REJECTION-ORIENTED LEARNING WITHOUT COMPLETE CLASS INFORMATION

Douglas de Oliveira Cardoso

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Felipe Maia Galvão França
João Manuel Portela da Gama

Rio de Janeiro
Março de 2017

REJECTION-ORIENTED LEARNING WITHOUT COMPLETE CLASS INFORMATION

Douglas de Oliveira Cardoso

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Felipe Maia Galvão França, Ph.D.

Prof. João Manuel Portela da Gama, Ph.D.

Prof. Carlos Eduardo Pedreira, Ph.D.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof. Guilherme de Alencar Barreto, D.Sc.

Prof. Rita Paula Almeida Ribeiro, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2017

Cardoso, Douglas de Oliveira

Rejection-Oriented Learning Without Complete Class Information/Douglas de Oliveira Cardoso. – Rio de Janeiro: UFRJ/COPPE, 2017.

XIII, 92 p.: il.; 29, 7cm.

Orientadores: Felipe Maia Galvão França

João Manuel Portela da Gama

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Referências bibliográficas: p. 81 – 90.

1. Clustering. 2. Open Set Recognition. 3. Data Streams. 4. Artificial Neural Networks. I. França, Felipe Maia Galvão *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Ouve teu pai, que te gerou, e não desprezes tua mãe, quando vier a envelhecer. Compra a verdade, e não a vendas; e também a sabedoria, a instrução e o entendimento. Grandemente se regozijará o pai do justo, e o que gerar um sábio, se alegrará nele. Alegrem-se teu pai e tua mãe, e regozije-se a que te gerou.
(Bíblia, Provérbios 23:22-25)

A meus pais, Rogerio e Rosana.

Acknowledgments

I am sincerely thankful to my advisors, professors Felipe França and João Gama. Working under their supervision since my MSc course (that is, over the last 7 years), this thesis is a milestone of this successful partnership. I would be honored to team up with you again in the years to come. Can you can put up with me any longer?

I am very grateful to those I met during my time in the labs I worked through the course of my PhD: LabIA-PESC, of COPPE institute of Universidade Federal do Rio de Janeiro; and LIAAD, of INESC TEC institute of University of Porto. I am a proud member of these two families, a LabIAnese and LIAADean. I was very fortunate to have labmates which truly helped to make my work easier and my life better during this period. I hope to give it back sometime, but I doubt can match your priceless support. I also thank the professors and other staff members of the just mentioned units, which also contributed to my academic formation.

The research which led to this thesis was financially supported by numerous institutions, which I enumerate next: the Brazilian government foundations CAPES (process 99999.005992/2014-01), CNPq, FAPERJ and FINEP; the European Commission through the project MAESTRA (grant number ICT-750 2013-612944); and the privately held companies INOVAX and General Electric.

My family is the base of everything I built so far in this life. There were so many lessons learned, guidance, relentless support, motivation and battles fought together. I am one with you. This accomplishment of mine is also yours. I express the ultimate gratitude to: my wife, Marceley, and our boy, Heitor; my parents, Rogerio and Rosana; my sister and her husband, Kenia and Carlos; my grandmother Ilda, and other grandparents (*in memoriam*) Ney, Nonô and Belinha; my father-in-law, Julio, mother-in-law Edelzia and brothers-in-law Hallan, Dandada, Nassor and Naomi; my uncles, cousins and other relatives. *Obrigado por tudo.*

I also have a word to my friends. There is a common saying in Brazil which states: "the one who has friends, has everything". I may not have numerous friends, but I know I can rely on the ones I have. Thank you for being by my side always, and for allowing me to have everything. I also include here my brothers and sisters in Jesus Christ from the churches Projeto Vida Nova de Jardim América, Vigário Geral, Xerém and Petrópolis, as well as Assembléia de Deus Novo Dia in Oporto.

At last, and above all, I want to thank God. In the Bible, Psalm 136:1 commands: "Give thanks to the Lord, for He is good. His love endures forever.". I have enjoyed Your goodness, grace and mercy day after day. I owe You every breath I take.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

APRENDIZADO ORIENTADO A REJEIÇÃO SEM INFORMAÇÃO COMPLETA DE CLASSE

Douglas de Oliveira Cardoso

Março/2017

Orientadores: Felipe Maia Galvão França
João Manuel Portela da Gama

Programa: Engenharia de Sistemas e Computação

Aprendizado de Máquina é comumente usado para apoiar a tomada de decisão em numerosos e diversos contextos. Sua utilidade neste sentido é inquestionável: existem sistemas complexos baseados em técnicas de aprendizado de máquina cujas capacidades descritivas e preditivas vão muito além das dos seres humanos. Contudo, esses sistemas ainda possuem limitações, cuja análise permite estimar sua aplicabilidade e confiança em vários casos. Isto é interessante considerando que a abstenção da provisão de uma resposta é preferível a cometer um equívoco ao realizar tal ação. No contexto de classificação e tarefas similares, a indicação desse resultado inconclusivo é chamada de rejeição. A pesquisa que culminou nesta tese proporcionou a concepção, implementação e avaliação de sistemas de aprendizado orientados à rejeição para duas tarefas distintas: reconhecimento em cenário abertos e agrupamento de dados em fluxo contínuo. Estes sistemas foram derivados da rede neural artificial WiSARD, que teve a modelagem de rejeição incorporada a seu funcionamento. Este texto detalha e discute tais realizações. Ele também apresenta resultados experimentais que permitem avaliar a importância científica e prática da metodologia de ponta proposta.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

REJECTION-ORIENTED LEARNING WITHOUT COMPLETE CLASS INFORMATION

Douglas de Oliveira Cardoso

March/2017

Advisors: Felipe Maia Galvão França
João Manuel Portela da Gama

Department: Systems Engineering and Computer Science

Machine Learning is commonly used to support decision-making in numerous, diverse contexts. Its usefulness in this regard is unquestionable: there are complex systems built on the top of machine learning techniques whose descriptive and predictive capabilities go far beyond those of human beings. However, these systems still have limitations, whose analysis enable to estimate their applicability and confidence in various cases. This is interesting considering that abstention from the provision of a response is preferable to make a mistake in doing so. In the context of classification-like tasks, the indication of such inconclusive output is called rejection. The research which culminated in this thesis led to the conception, implementation and evaluation of rejection-oriented learning systems for two distinct tasks: open set recognition and data stream clustering. These system were derived from WiSARD artificial neural network, which had rejection modelling incorporated into its functioning. This text details and discuss such realizations. It also presents experimental results which allow assess the scientific and practical importance of the proposed state-of-the-art methodology.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Because of technological facts of our time as social networks, Internet of Things, ubiquitous sensing and others, data generation processes became faster and more numerous. At the same time, the prior availability of labeled observations is now much less common or guaranteed than before. Thus, while such abundance of data could be considered ideal for automated computational learning, it still could have pros and cons.

Despite some possible drawbacks, having more data at hand is generally positive. In this regard, some training data which is unlabeled can be not only useful but necessary for the accomplishment of some machine learning tasks. This helps to understand the importance of the two problems addressed during this research: open set recognition (SCHEIRER *et al.*, 2013) and data stream clustering (GAMA, 2010). Both concern knowledge obtainment from observations despite not knowing the ground truth regarding their classes.

Open set recognition is a classification-like task: its accomplishment requires the identification of observations which belong to some modeled classes, named *targeted* classes. Additionally, this task also considers the existence of classes in the problem domain besides the targeted ones. Observations from these non-targeted classes should be rejected. That is, rejection means to avoid ruling an observation as an element of any of the targeted classes. The need for proper handling of elements of classes beyond those of interest is frequently ignored, even in works in the literature. This leads to the improper development of learning systems, which may obtain misleading results when evaluated in their test beds, consequently failing to keep the performance level while facing some real challenge.

Clustering is one of the most popular subjects of machine learning literature, addressed in numerous works through the years. Comparatively, its batch version is significantly different from its realization feeding from a data stream: the first can be described as trying to determine clusters which represent the entire input sample, as a single process; the last leads to the definition and continuous update

of a set of clusters which reflects data current state, considering how observations are ordered and temporally related. A basic requirement for data stream clustering is to sensibly associate most recent data to current clusters. Alternatively, it is also necessary to perceive the absence of such relation, what could be interpreted as some sort of rejection.

Facing these challenges, WiSARD (ALEKSANDER *et al.*, 1984) was brought into play as powerful, flexible, multi-purpose learner. This artificial neural network model provides the means for pattern recognition working as a lazy learner, memorizing and matching parts of its inputs. The original and most frequent use of this model is standard classification. However, it has been used for other tasks: unsupervised learning (WICKERT e FRANÇA, 2001), rule induction (COUTINHO *et al.*, 2014), generative modeling (GRIECO *et al.*, 2010) and natural language processing (CARNEIRO *et al.*, 2015; DE CARVALHO *et al.*, 2014) are some recent examples of those. Despite such variety, the exploration and analysis of WiSARD for rejection-oriented learning was never performed before this research.

1.1 Research Aims and Objectives

This research originally aimed the development of an approach to data stream clustering which would overcome limitations of existing alternatives regarding data aging management: a finer control over the influence of past data on current clusters was desired. After a preliminary analysis, WiSARD was considered an appropriate tool to support the accomplishment of such goal. However, during an early stage of this work, the perspective of the sketched WiSARD-based clusterer with respect to rejection prompted a closer inspection of the base model in this sense. Consequently, this allowed to properly include open set recognition in the scope of this research, as a possibly simpler task in which rejection modeling also played a major role.

Based on the just provided abstract definition of the two research aims, the following concrete objectives were established:

Identify WiSARD rejection-friendly features.

It was necessary to verify which characteristics of the standard WiSARD would support the development of the targeted rejection-capable variation of this model. Such initial step also helped to foresee some challenges to overcome later in this research.

Qualitatively compare WiSARD to existing rejection tools.

Analyzing how WiSARD differs from other methods which could also be used for rejection allows to better perceive strengths and weaknesses of the methodology being developed. This way, the extent of the contributions of this work

could be defined more clearly. This analysis could also suggest some ideal conditions for application of the methodology.

Outline a stream-oriented information disposal mechanism for WiSARD.
It is necessary to alter how WiSARD manages stored knowledge: information pieces should be temporally organized, as precisely as possible; this way, they could be discarded as they become older. Such action has to occur in parallel with stored knowledge update and increment which happen during stream processing. Therefore, its efficiency is as important as its effectiveness.

Adapt the proposed open-set rejection criteria to clustering.
Open set recognition is a supervised learning task, in which rejection criteria can be adjusted based on training data labeling. The same is not true for clustering, what disallows a straightforward translation from one context to the other. This prompts the definition of a more specific strategy to accomplish clustering-oriented rejection.

Implement WiSARD-based systems for the targeted tasks.
This objective represents the conception of two systems: the aforementioned variation of WiSARD, embedding rejection modeling into its operation, to be used for open set recognition; another rejection-capable derivation of WiSARD which should be an unsupervised, stream-oriented learner. For validation purposes, this objective also requires to experimentally evaluate the developed systems, as well as to report and discuss the obtained results.

1.2 Thesis Outline

The remainder of this text is organized in 4 chapters. First, Chapter 2 introduces the building blocks of this work: it starts with a description of artificial neural networks, explaining their biological inspiration and basic concepts; then, the WiSARD model is presented, what includes comments about its features, operation and the intuition behind its functioning; at last, there are overviews of open set recognition and data stream clustering, providing the definition of these problems, key concepts and terms, works in the literature concerning them, and how they can be related.

Next, Chapter 3 presents the developed approach for open set recognition. The ideas which support such accomplishment are explained as well: in first place, the interpretation of WiSARD matching operation as an observation-to-sample proximity meter; also, the computation and use of rejection thresholds which were embedded into WiSARD functioning. The proposed system was evaluated through a collection of experiments, whose descriptions and results are detailed and commented.

Chapter 4 follows, showing how the just-discovered WiSARD rejection power could also be explored in the context of data stream clustering. In this case, dynamic rejection criteria had to be defined, reflecting clusters evolution during stream processing. Besides explaining the conception of such criteria, this chapter also details how the base learning model was modified to perform online learning, by the disposal of outdated knowledge. Like the previous chapter, this one also ends with the experimental evaluation of the proposed approach, followed by final remarks.

At last, Chapter 5 summarizes all the stages of this research journey, and the decisions which led to each of them. This chapter also highlights the most noteworthy findings of this work, considering novelty and relevance. Suggestions of unprecedented scientific explorations which would be related to this research are also given, targeting to direct future works.

Chapter 2

Research Background

For a better understanding of the accomplishments of this work, an overview of the concepts on which it was built is indispensable. Moreover, to establish the relation to other works is especially important here: the covered themes are of great interest of the research community, and were addressed abundantly in the literature. Despite this, there is still room for improvements and alternative approaches, as unprecedented challenges emerge from the review of some classical research targets under novel perspectives.

Data stream clustering, one of the main subject of this work, is an example of these targets. That is, clustering is one of the most basic machine learning tasks, and was analyzed taking into consideration a great variety of premises. Still in this regard, to cluster a data stream instead of a data set, the most common setup, is an alternative point of view of the same matter. Among numerous derivations of the basic clustering task, the one analyzed in this research is remarkable for the great number of real applications it covers.

Similarly to data stream clustering and batch clustering, open set recognition is strongly related to classification. However, the first requires handling data extraneous to all known classes in an exclusive manner, while this is not necessary to perform the last. As it is shown later in this thesis, this proximity between these two tasks led to questionable problem modelings found in other works, which describe the inadvertent use of classifiers when extraneous data should be taken into account. Extraneous data is what separates open set recognition from classification, while it also makes data stream clustering and open set recognition closer to each other.

The base of the solutions described in this thesis is an artificial neural network model which was conceived for classification. Through the course of this research some uncommon properties of this model were explored broadening its use to the tasks approached. Therefore, it is presented a comparison of this model to other artificial neural network models focusing on these distinctive characteristics.

This is how this chapter is organized: Section 2.1 provides a brief summary of the

current knowledge regarding artificial neural network models; Section 2.2 details the memory-based artificial neural network model which was further developed during this research; the definition of open set recognition is presented in Section 2.3, as well as a comparison to classification; at last, Section 2.4 introduces the concepts and assumptions concerning data stream clustering used through this work.

2.1 Artificial Neural Networks

Artificial neural networks are statistical tools whose design was inspired in nervous systems of living beings, created to emulate some of the learning capability of their biological counterparts. There exists a great variety of artificial neural network models, which have different characteristics and are used for several purposes: function approximation, signal processing, classification, clustering, time series prediction and others. But all these models share a basic design principle: each of them is defined as a collection of units, called nodes or neurons, which are combined according to the model definition, working collectively.

2.1.1 Biological Analogy

Biological neurons operate as signal processing units: they receive stimuli through its dendrites, which are organized as a tree; these stimuli are combined during the traversal of the dendritic tree; resulting signals of such combination reach the soma, where a response for such inputs is generated; this response is forwarded trough the axon to muscles, glands or other neurons whose dendrites are connected to this axon by synapses. Figure 2.1 indicates these components of a generic neuron.

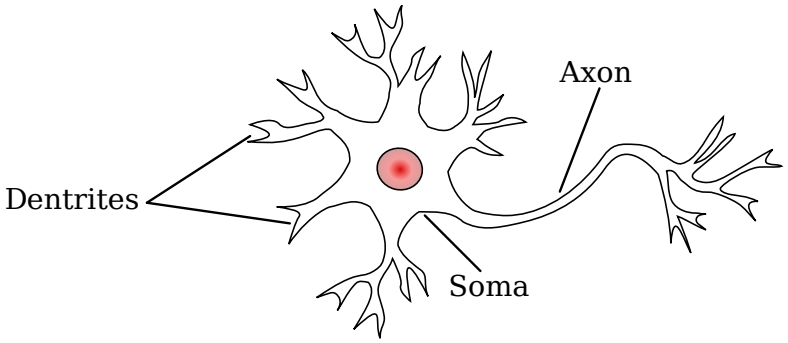


Figure 2.1: A sketch of a biological neuron. *

The most popular mathematical abstraction of biological neurons was originally proposed by MCCULLOCH e PITTS (1943). In such model, the synapses are

*Adaptation of image licensed under Creative Commons Attribution-Share Alike 3.0 Unported. Source: http://en.wikipedia.org/wiki/Neuron

substituted by edges, connecting the nodes of the neural network. The stimuli the neuron receives is substituted by the input of numerical values. These values are multiplied by numerical weights associated to the edges they traverse. At last, the sum of these multiplications is input to some function, whose outcome is used as the output of the neuron. Such modeling is reasonable from both biological and mathematical points of view.

2.1.2 Practical Use

From a generic definition of a single neuron, the functioning of a system of those units can be analyzed. An example of artificial neural network is shown in Fig. 2.2. It is an instance of the vastly used Multilayer Perceptron model. Here it is a list of some of its model-specific characteristics:

- there is no cycle in the network;
- the nodes are organized in layers, which are totally ordered;
- a node in a layer is linked to all nodes of the following layer;
- every node applies the same function on the summation of its inputs.

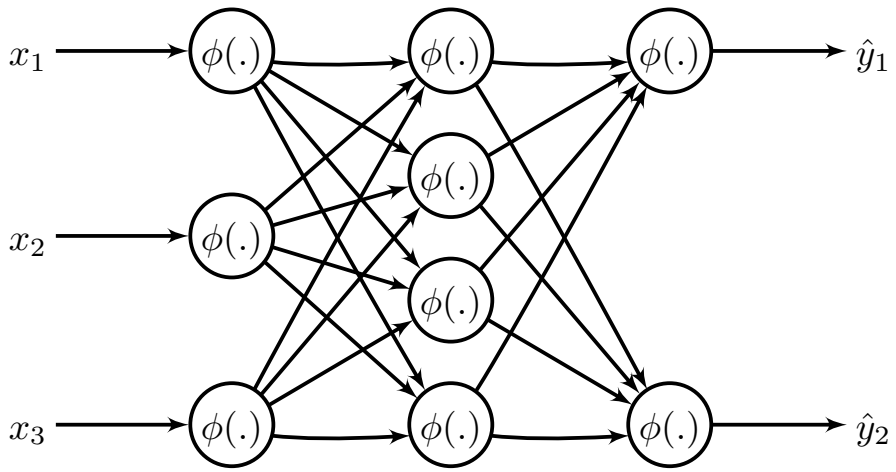


Figure 2.2: A hypothetical artificial neural network instance.

The number and type of parameters used to define a network also varies according to the model being considered. The number of nodes of the network is the most universal parameter: it is usually a network architecture option, defined depending on how complex are the concepts to be learned. For layered networks, the number of nodes and the number of layers are jointly defined by the designation of the number of nodes per each layer. Another common parameter is the function applied by

each node on its inputs, named *activation function*. Some popular choices are: the logistic function, the hyperbolic tangent, the rectifier and radial basis functions.

What the network depicted in Fig. 2.2 does is to transform a given input vector $\mathbf{x}$ according to the weights of its edges and the function applied by its nodes, producing an output vector $\hat{\mathbf{y}} = \hat{f}(\mathbf{x})$. Here, to learn means to produce an output vector $\hat{\mathbf{y}}$ for a given input vector $\mathbf{x}$ as close as possible to $\mathbf{y} = f(\mathbf{x})$, the true corresponding output to input $\mathbf{x}$. This is pursued by the adjustment of the weights of the edges, the only modifiable parts of the network.

In most practical situations, there is a collection of pairs $(\mathbf{x}_i, \mathbf{y}_i)$, the training sample, and it is desired to obtain a network whose error (that is, the difference between the provided and the desired outputs) is as small as possible. This is an optimization problem, usually approached using the backpropagation algorithm, which is briefly described in Algorithm 2.1. The algorithm works combining the influences of each observation on network state, altering it gradually until being unable to improve its condition. This optimization can get stuck in a state which is the best of its neighborhood, but not globally. Despite the lack of guarantee that the best possible state will be found, the Multilayer Perceptron model combined with the backpropagation algorithm is, deservedly, one of the most used machine learning tools, because of its mathematical soundness and practical success.

1: Set the weights of the edges to random values
2: **repeat**
3: **for all** training pairs $(\mathbf{x}_i, \mathbf{y}_i)$, considered in some random order **do**
4: Compute $\hat{\mathbf{y}}_i$, using $\mathbf{x}_i$ as an input to the network
5: From last to first, calculate the contribution of each edge to error $\mathbf{y}_i - \hat{\mathbf{y}}_i$
6: Adjust the weights of the edges, according to their contribution
7: **until** the network state is considered stable

Algorithm 2.1: A simplified description of the backpropagation algorithm.

The most popular artificial neural network models rely on the modification of weights of its edges by the superposition of the effects of the observations which compose the training sample. By this mechanism, the knowledge extracted from data can not be updated straightforwardly: if some observations are added or removed from the training sample, the network state which minimizes the error may change substantially. This fact hampers the use of artificial neural networks in data stream mining tasks (MENA-TORRES e AGUILAR-RUIZ, 2014). Although such applications can be found in literature (PAVLIDIS *et al.*, 2011; RODRIGUES e GAMA, 2009; SILVA e MARQUES, 2012), none has a precise control over knowledge update: after an observation is used to change the current state of the network, its influence becomes permanent, never being truly canceled afterwards. Moreover,

depending on learning rate setting, they may be subject to catastrophic forgetting (FRENCH, 1999), being over-sensitive to new information. This problem is also known as the stability-plasticity dilemma, which was addressed by some works in the literature (CARPENTER *et al.*, 1991, 1992; GROSSBERG, 1982).

2.1.3 Weightless Modeling

A naive way to extract and store some information regarding a data set is to memorize it. This alternative to the learning mechanism just described is very intuitive, if not obvious. Moreover, this strategy has at least one indisputable advantage over the weights-adjustment approach: it ensures that the error over the training sample is as small as possible, zero. It also copes perfectly with modifications of the training sample: the addition and removal of handpicked observations is possible, as they are stored apart.

On the downside, the first issue the consideration of this approach raises regards its feasibility: is it possible, from a practical point of view? This question concerns two different challenges: to store a possibly enormous amount of information, according to the dimensionality and length of the data set; to organize such observations so that their retrieval is sufficiently efficient.

Another important inquiry about this mechanism is the following: how does it generalizes the knowledge obtained from the training sample? In other words, if an observation $\mathbf{x}$ which is not part of the training sample is input, how does it provides a reasonable prediction $\hat{\mathbf{y}}$ with respect to its true corresponding output $\mathbf{y}$? It can be noted that the previously described weights-based network is natively capable of performing such generalization.

An action which address these two points (feasibility and generalization) is to map the input space to a set of smaller cardinality. First, with respect to feasibility, this reduces the number of possible input vectors, making their storage and retrieval easier. This also implies some sort of generalization: some distinct elements in the original space (i.e., the domain) would be transformed into the same element in the image of the mapping; this way, an observation which do not belong to the training sample could be mapped to an element to which an observation in the training sample was previously mapped to. For example, considering $\mathbf{x} \in \mathbb{R}$, a rounding function ($r\colon \mathbb{R} \to \mathbb{Z}, r(x) = \lfloor x \rceil$) could be used to perform such mapping.

However, the use of a mapping as just detailed allows different input vectors of the training sample to be transformed into the same element in the image. If their corresponding output vectors are different, it is impossible to assure the absence of error on learning the training sample. On the other hand, if these vectors are equal, to replicate the removal of observations from the training sample it is necessary to

count how many input vectors in the training sample were mapped to each element in the image.

Despite drawbacks as those just described, the combination of memorization and mapping is viable base of a learning system. Weightless artificial neural networks (ALEKSANDER *et al.*, 2009) are memory-based alternatives to weights-based ones. As the name implies, all links of these networks have no weight, acting as the simplest communication channels, exercising no effect on data traffic. Therefore, their nodes are responsible for the learning capability these networks exhibit. These nodes operate as memory units, keeping small portions of information, which are combined when a query regarding the knowledge the system possesses needs to be answered. These information pieces are the outcome of mapping the data used as knowledge source.

The biological inspiration of these nodes is the influence of dendritic trees on neuron functioning. In the abstraction of MCCULLOCH e PITTS (1943), such trees were modeled as a weighted edges, which multiply the neuron inputs before the application of the activation function on their summation. Although practical, this is a rough simplification of how these trees operate. As a matter of fact, the input signals of biological neurons, which can be of two types (excitatory or inhibitory), are combined by the dendritic tree before reaching the neuron soma, where they prompt the generation of a new signal. This action can be naturally compared to the definition of a boolean key used to access an index of boolean values, which is how the most basic neurons of weightless artificial neural network models work.

Before detailing a weightless artificial neural network model in the Section 2.2, Table 2.1 summarizes the differences between networks on opposite sides with respect to using the adjustment of the weights of the edges as a learning mechanism.

Aspect	Weights-based	Weightless
Knowledge storage	Edges weights	Network nodes
Learning process	Iterative adjustment	One-shot memorization
Generalization	Inherent	Mapping-derived

Table 2.1: Differences between artificial neural networks regarding weights.

2.2 WiSARD

The WiSARD (**Wi**lkes, **S**tonham and **A**leksander **R**ecognition **D**evice) is a weightless artificial neural network model (ALEKSANDER *et al.*, 1984). The way it works is arguably the simplest among all artificial neural network models: it implements the already described mapping-and-memorization scheme in a collection of nodes

organized in a single layer; the outputs these nodes can provide are limited to the values 0 and 1; these outputs are aggregated through ordinary summation. In spite of its simplicity, WiSARD is an effective learning model.

This model was created to be used in classification tasks. Classification can be seen as an specialization of the general action of learning which is being discussed so far in this chapter: the generic target is to provide some meaningful prediction $\hat{\mathbf{y}}$ about the true corresponding output $\mathbf{y}$ of a given input $\mathbf{x}$, based on the knowledge extracted from a sample of pairs $(\mathbf{x}_i, \mathbf{y}_i)$; for classification, y is unidimensional (for this reason, y will be used instead of $\mathbf{y}$ in this context), the number of possible values of y is finite (in general, a few hundreds at most), these values are known a priori and they are called classes.

Now consider a hypothetical binary (i.e, two classes) classification task. A system like a Multilayer Perceptron network is adapted to classification by the association of the values +1 and -1 to each class. These classes will be referred to as the positive and the negative class, respectively. After training, it is expected that such system outputs a positive value if the input observation is of the positive class, and the opposite for the negative class. Thus, this system is mathematically described as a function $\hat{f}\colon \mathbb{R}^d \to \mathbb{R},\ \mathbf{x} \mapsto y$, and the prediction it provides is $\hat{y} = sgn(\hat{f}(\mathbf{x}))$.

On the other hand, WiSARD provides for each class a value in the interval $[0, 1]$: in the current example, a binary classification task, it would be two values. The value concerning a class represents how well the provided observation matches the acquired knowledge regarding that class. It is straightforward to transform a two-values answer provided by WiSARD to a real number as an output from a Multilayer Perceptron: the subtraction of the two given values is enough for this. However, the answer format of WiSARD is more informative: for example, a small difference between two values close to 1 possibly is an evidence that both classes could be the true class of the input observation; but the same difference between values close to 0 could be an evidence that none of the classes are good guesses for the true class.

The values which compose an answer obtained from WiSARD are computed from structures called discriminators. Each discriminator is responsible for storing the knowledge regarding a class, as well as assessing the matching between the class it represents and any observation whose true class has to be predicted. How a discriminator learns about its respective class is described in Algorithm 2.2. In a sentence, it records in its nodes the values resulting from mapping the observations in the training sample. Mind some notation introduced here: the discriminator of class $\dot{y}$ is represented by $\Delta_{\dot{y}}$; the j^{th} node of $\Delta_{\dot{y}}$ is represented by $\Delta_{\dot{y},j}$; the number of nodes which compose each discriminator is represented by δ.

There are several analogies between hardware systems and WiSARD. Consequently, some parts of its structure are named using terms which belong to this

1: **for all** $\Delta_{\dot{y},j}$, the network nodes **do**
2: $\quad \Delta_{\dot{y},j} \leftarrow \varnothing$ ▷ All nodes operate as sets, and are initially empty
3: **for all** pairs $(\mathbf{x}_i, y_i)$, the training sample **do**
4: $\quad$ Let addressing$(\mathbf{x}_i) = (a_1\ a_2\ \cdots\ a_\delta)$ be a vector of values of mappings of $\mathbf{x}_i$
5: $\quad$ **for all** addresses a_j in addressing$(\mathbf{x}_i)$ **do**
6: $\quad\quad \Delta_{y_i,j} \leftarrow \Delta_{y_i,j} \cup \{a_j\}$ ▷ Adding address a_j to node $\Delta_{y_i,j}$

Algorithm 2.2: A description of WiSARD training procedure.

domain. For example, its nodes are called RAM nodes, a direct reference to their memory-like operation, different from the functional nodes of the weights-based networks. Like physical RAM modules, their content is retrieved or altered using addresses, defined by an *addressing* procedure. Despite this nomenclature, RAM nodes work identically to sets, well-known mathematical structures, and are commonly implemented using hash tables. Likewise, addresses can be seen as simple vectors, obtained from mapping the observations.

After training, a WiSARD instance can rate the matching between any known class $\dot{y}$ and an observation $\mathbf{x}$ as shown in expression (2.1a). At last, an observation $\mathbf{x}$ is classified according to expression (2.1b).

$$\text{matching}(\mathbf{x}, \dot{y}) = \frac{1}{\delta} \sum_j [\text{addressing}_j(\mathbf{x}) \in \Delta_{\dot{y},j}]^\dagger \quad ; \tag{2.1a}$$

$$\hat{y} = \underset{\dot{y}}{\text{argmax}}\ \text{matching}(\mathbf{x}, \dot{y}) \quad . \tag{2.1b}$$

There is an extra level of generalization implied by the matching computation. Consider that a discriminator Δ_+ was trained using the observations of a set $X_+ = \{\ldots, \mathbf{x}_i, \ldots\}$. For a given observation $\mathbf{x}$, expression (2.2) holds: $\mathbf{x}$ perfectly matches Δ_+ (i.e, matching$(\mathbf{x}, +) = 1$) iff all addresses of $\mathbf{x}$ match addresses of observations in X_+. Thus, the combination of addresses obtained from different observations allow the recognition of observations which do not belong to the training sample.

$$\forall_i, \exists_{\mathbf{x}' \in X_+}\ \text{addressing}_i(\mathbf{x}') = \text{addressing}_i(\mathbf{x}) \iff \text{matching}(\mathbf{x}, +) = 1 \quad . \tag{2.2}$$

Figure 2.3 presents the structure of a WiSARD system for binary classification. This illustration complements the model overview which was just provided.

[†]The Iverson bracket: $[L] = 1$ if the logical expression L is true; otherwise, $[L] = 0$.

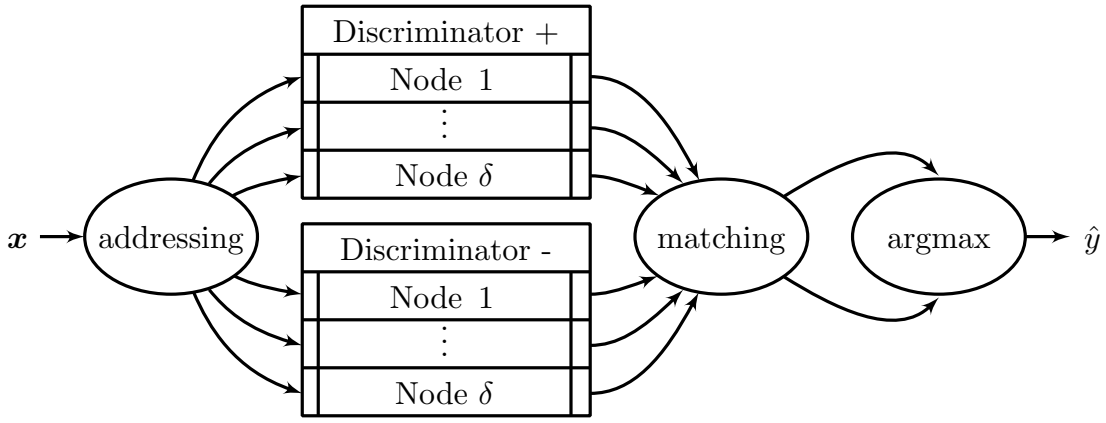


Figure 2.3: The scheme of a hypothetical WiSARD classifier.

2.2.1 Addressing

An important part of WiSARD was not detailed yet: the addressing procedure. So far this is a black-box process which receives an observation as input and outputs a vector of values to be registered by the RAM nodes. This operation plays a major role in learning effectiveness: it is responsible for the definition of every small portion of knowledge kept by the nodes. Even the network structure is bonded to addressing, as the number of nodes in each discriminator is the same as the number of addresses generated by this procedure. For such reason, this section is dedicated to further explain this component of the system.

A proverbial rule which is a practical guide for classification is the "duck test":

> "If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck."

To 'look like a duck', 'swim' and 'quack' are features of an animal that could be rationally classified as a duck. If another animal has just one of these three features, it would be reasonable to be less sure about its classification as a duck. Thus, the number of features an element (animal) has in common with other acknowledged class members (ducks) is a measure of how sensible it is to relate that element to that class. While the evaluation of such number is equivalent to the matching$(\mathbf{x}, \dot{y})$ computation, the features to be used for such computation result from addressing.

Now consider a binary classification task regarding black-and-white, $m \times n$ images. Let T denote the training sample. Let T^+ and T^- denote the respective subsets of T containing observations of the positive and negative class exclusively. Let $\mathbf{A}_{m \times n} = [a_{ij}]$ be an image which should be classified.

A row of pixels of $\mathbf{A}$ could be used as a feature for its classification: for example, if no image in T^+ has the same first row as $\mathbf{A}$, while such accordance can be found with respect to T^-, this can be considered an evidence that $\mathbf{A}$ belongs to the negative

class. In fact, it is possible to obtain such evidences using any subset of the images: a row, a column or any collection of pixels can be used for this. This way, an acceptable definition of addressing for this image classification task follows:

1. Let $c_k = (\cdots \ (i,j) \ \cdots)$ be a collection of pixel coordinates;
2. Let $\Pi_{c_k}(\mathbf{A}) = (\cdots \ a_{i,j} \ \cdots)$ be a projection of $\mathbf{A}$ regarding c_k;
3. Finally, addressing$(\mathbf{A}) = (\Pi_{c_1}(\mathbf{A}) \ \cdots \ \Pi_{c_\delta}(\mathbf{A}))$.

For a better understanding, Fig. 2.4 shows how the projection $\Pi_{c_k}(\mathbf{A})$ works. In this example, $c_k = ((1,1) \ (2,2) \ (3,3) \ (4,4) \ (5,5))$ and the values 0 and 1 are associated to white and black (gray) pixels respectively. The result of the projection is an address, in the form of a bit (i.e., 0-1) vector.

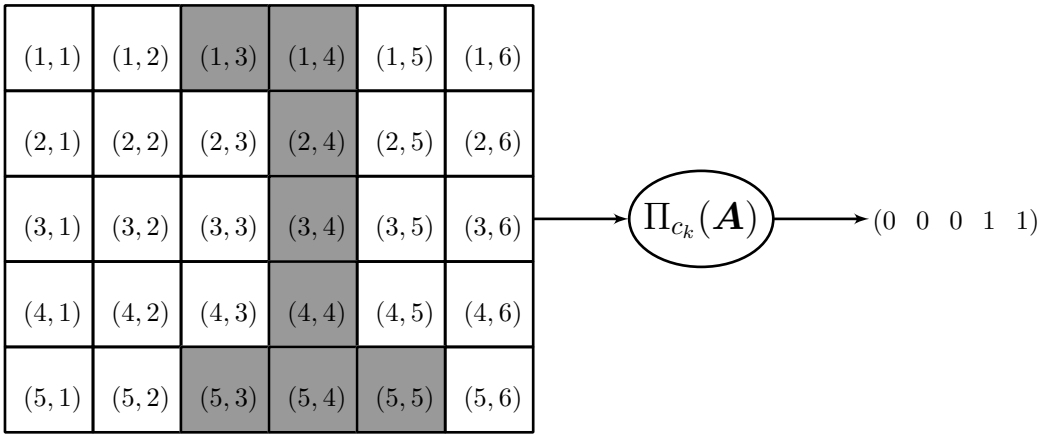


Figure 2.4: A projection of a black-and-white image.

How the collections of pixel coordinates $c_1, \ldots, c_\delta$ are defined is the natural follow-up question. There are two important aspects in this regard. The first one is their size: a c_k could refer to a single pixel as well as to all mn pixels. The amount of information a sole pixel brings is the smallest possible, what likely reduces its usefulness as a feature: to decide if an animal is a duck or not, it is useless to know just that the animal has feet, because this is not a distinctive characteristic. On the other hand, to use all pixels as a single feature harms knowledge generalization: to decide if an animal is a duck or not, it is useless to know that the animal is not a perfect clone of a previously known duck. Therefore, the size of c_k should balance the lack and the excess of information it may lead to.

Still about the size of $c_1, \ldots, c_\delta$, remember expression (2.1a): it shows that each node $\Delta_{\dot{y},i}$ has the same weight on matching$(\mathbf{x}, \dot{y})$. This is an indication that the size of each different c_k should be the same: a positive output of a node whose addresses are mn-dimensional vectors is a stronger classification evidence than that of a node with 1-dimensional addresses; as the lengths of the addresses are not taken into

account in expression (2.1a), the collections of pixel coordinates which determine these addresses should have the same size.

The second aspect of the definition of $c_1, \ldots, c_\delta$ is the criterion used for this. From a human point of view, the most natural collections of pixels are rows, columns and rectangular windows. However, the combination of disjoint pixels in a feature generally avoids the redundancy related to neighboring pixels. Moreover, pixels apparently unrelated can define unexpectedly interesting features. For these reasons, the collections of pixel coordinates are defined randomly, but guaranteeing that each pixel coordinate is an element of exactly one of the collections, as described by expression (2.3).

$$c_i \cap c_j = \varnothing, \; i \neq j \quad . \tag{2.3}$$

As each of the mn pixel coordinates is used only once and δ, the number of nodes per discriminator, is a parameter of the model, the length of the addresses of the nodes is fixed: $\beta = mn/\delta$. This way, the parametrization of a WiSARD instance can described by the declaration of either δ or β. If mn is not divisible by a desired δ (or β), the use of addresses with different lengths becomes inevitable: this situation is acceptable, but this difference should be as small as possible, according to what was explained in the second previous paragraph.

For a practical understanding of how WiSARD parametrization, addressing and matching work together, Fig. 2.5 and Table 2.2 show an example of their combination. In this hypothetical situation, there are two observations which belong to the positive class and a third observation $\mathbf{A}$ which is used to compute matching$(\mathbf{A}, +)$. This computation is performed for different definitions of each collection of pixel coordinates c_i used for addressing. As a didactic example, human-friendly collections of pixel coordinates, as row and column groups, were used.

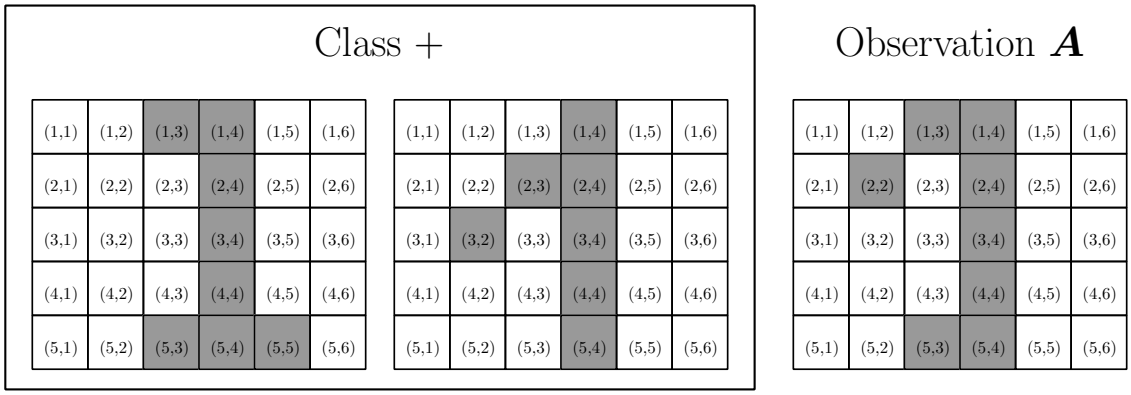


Figure 2.5: Observations of the didactic example of WiSARD operation.

It can be noticed the relation between the β, the length of the addresses and the resulting matching rate: it is harder for the addresses to coincide when they are lengthier, what leads to smaller matching rates. Nevertheless, in this example the

β	δ	Addressing c_i setup	matching$(\mathbf{A}, +)$
1	30	$c_i = i^{th}$ pixel	$29/30 = 96.6\%$
5	6	$c_i = i^{th}$ column	$5/6 = 83.3\%$
6	5	$c_i = i^{th}$ row	$3/5 = 60.0\%$
10	3	$c_i = i^{th}$ column pair	$2/3 = 66.6\%$
15	2	$c_i = i^{th}$ column trio	$1/2 = 50.0\%$
30	1	$c_1 =$ all pixels	$0/1 = 00.0\%$

Table 2.2: Parameter and addressing definitions, with respective matching rates.

matching rate for $\beta = 10$ (66.6%) is still higher than that of $\beta = 6$ (60%). Thus, the relation between β and matching is not purely deterministic.

2.2.2 Binarization

In Section 2.2.1 it was shown how to use WiSARD to classify black-and-white images. It could be noticed that the addresses obtained from binary observations, as the 0-1 matrices induced by black-and-white images, are bit vectors. The use of such vectors as addresses is a basic characteristic of WiSARD: the addresses of physical RAM modules are routinely declared as bit vectors; in this format, the length of the addresses is directly related to the amount of information each node handles and the generalization level of the classifier.

For images other than black-and-white ones, there are various binarization methods specific to such inputs (CHAKI *et al.*, 2014): for example, the transformation of a grayscale image can be performed by some sort of thresholding (SEZGIN e SANKUR, 2004); a color image can be converted to black-and-white directly or to grayscale as an intermediate step. Most of these methods were developed for image segmentation, targeting to provide simpler descriptions of images while preserving to a certain extent the information they bring, what is interesting from a machine learning perspective. By the application of such techniques, WiSARD and other weightless artificial neural network models were successfully used in diverse computer vision tasks (JUNIOR *et al.*, 2015; NASCIMENTO *et al.*, 2015; STAFFA *et al.*, 2015).

Despite this strong bond between WiSARD and image-related tasks, the application of this model is not limited to this context. Any classification task can be approached using WiSARD as far as a proper binarization method for the observations is available. That is, a method to transform observations in its original format to a binary format preserving structural characteristics of the data, as the pairwise distance. This description resembles that of dimensionality reduction (VAN DER MAATEN *et al.*, 2007) techniques as multidimensional scaling and locally linear em-

bedding. However, it is noticeable that while these just mentioned techniques target to describe data using less dimensions, the binarization procedures for WiSARD usually increase the dimensionality of data to compensate the poorer representativeness of binary dimensions compared to integer or real ones.

The most popular data representation is the real-valued feature vector, which is the input format of models like the Multilayer Perceptron: $\mathbf{x} \in \mathbb{R}^d$. There exists various methods to transform this kind of input to a WiSARD-friendly input (KOLCZ e ALLINSON, 1994; LINNEBERG e JORGENSEN, 1999). One of the simplest and most widely used alternatives is the unary encoding, also known as bar chart or thermometer encoding.

A mapping $u : [0, 1]^n \to \{0, 1\}^{n \times \gamma}$ is a mathematical interpretation of an unary encoding. The output matrix of such mapping has one row for each dimension of the input vector. The number of columns is defined according to γ, a parameter which represents the desired resolution of the values which compose the input vector. Algorithm 2.3 details the unary encoding procedure.

Let $\gamma \in \mathbb{N}$ be the desired resolution of the values
procedure UNARYENCODING($\mathbf{x} \in [0, 1]^n$)
    Let $\mathbf{A} = 0_{n \times \gamma}$ be a $n \times \gamma$ zero matrix
    for all x_i **do**
        $v \leftarrow \lfloor \gamma \; x_i \rceil$
        for all $j \in \{1, 2, \cdots, v\}$ **do** ▷ From 1^{st} to v^{th} column
            $a_{ij} \leftarrow 1$
    return A

Algorithm 2.3: Binarization of real-valued vectors using unary encoding.

Graphically, an unary encoding produces a black-and-white horizontal bar chart which naturally refers to $\mathbf{x}$ in its original representation. Intuitively, the similarity between elements is preserved. Figure 2.6 illustrates this fact.

Indeed, increasing γ, the Hamming distance (the number of different entries) between the unary representations $\mathbf{A}_1$ and $\mathbf{A}_2$ of any pair of observations $\mathbf{x}_1$ and $\mathbf{x}_2$ comes closer to the L_1 distance between these elements in their original representation, proportionally. Expression (2.4) mathematically describes this statement, for some constant $k \in (0, \infty)$. For classification, however, the smaller γ which allows to correctly distinguish observations of different classes should be used: this favors knowledge generalization and reduces computational overhead.

$$k \underbrace{\sum_i |x_{1i} - x_{2i}|}_{L_1 \text{ distance}} = \underbrace{\sum_{i,j} |a_{1ij} - a_{2ij}|}_{\text{Hamming distance}} \text{ , for a sufficiently large } \gamma \text{ .} \tag{2.4}$$

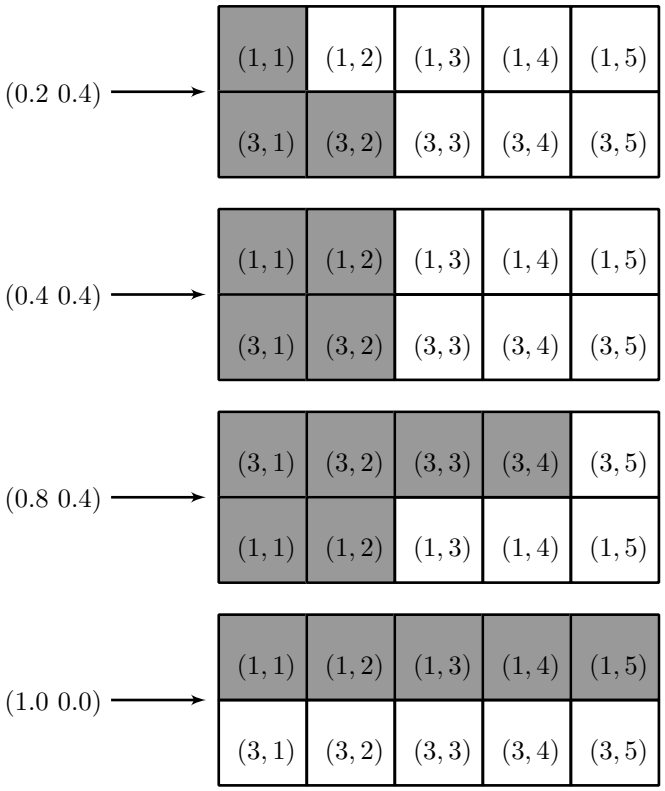


Figure 2.6: Real-valued feature vectors and their unary counterparts, with $\gamma = 5$.

2.2.3 Saturation

When designing a solution to a learning problem, a fundamental matter in this regard is the complexity of the concept to be learned. Different problems require the use of different tools, with different adjustments. A system like the Multilayer Perceptron works explicitly fitting to the training data a function it defines, targeting to reproduce the underlying data generation process. Thus, the defined function should conform with this process. Two undesired situations which can happen to systems of this kind are:

Underfitting

The system is too simple to learn the target concept, providing a too rough approximation of the data source;

Overfitting

The system is too complex, leading to an approximation based on over-complicated criteria inferred from the data.

WiSARD is not based on this principle of function fitting, what makes it natively free of these problems. In spite of this, a WiSARD classifier still needs to be properly

set up with respect to the complexity of the concepts to be handled. This happens by the adjustment of β, the length of the addresses used to input and retrieve the contents of the RAM nodes. This parameter establishes the minimum amount of information two observations must have in common to be possibly related by the classifier. This way, the chance of associating an observation being classified to an observation in the training sample (and, consequently, to its class) is smaller with a higher β. If the concept is complicated, it is sensible to be more strict about the generalization of the knowledge obtained from data, requiring a clearer resemblance to some observation of the training sample to make any inference based on it.

Although the use of a too high β harms the generalization capability of a WiSARD classifier, the consequence of defining a too low value for this parameter is even worse: the discriminative power is lost, as it becomes detrimentally easy to relate the observation being classified to observations belonging to any of the known classes. In other words, the classifier may not be able to rightfully decide which class best matches the observation in question, because by mere casualty this observation has β bits in common with observations of any class.

A simple example comes in handy. Consider the observations depicted in Fig. 2.7, and $\beta = 1$ (this way, the number of neurons $\delta = 30$). After training, the observation $\mathbf{A}$ perfectly matches the positive class (matching$(\mathbf{A}, +) = 1$) although there is no apparent similarity between $\mathbf{A}$ and any observation of the positive class. This happens because all 30 neurons, which work with 1-bit addresses, recorded the occurrence of all possible addresses during training: 0 (from the first positive observation) and 1 (from the second one). Therefore, the response of these neurons to any query regarding their content would be positive. Consequently, not only $\mathbf{A}$ but any observation, with no exceptions, would perfectly match the positive class in this situation.

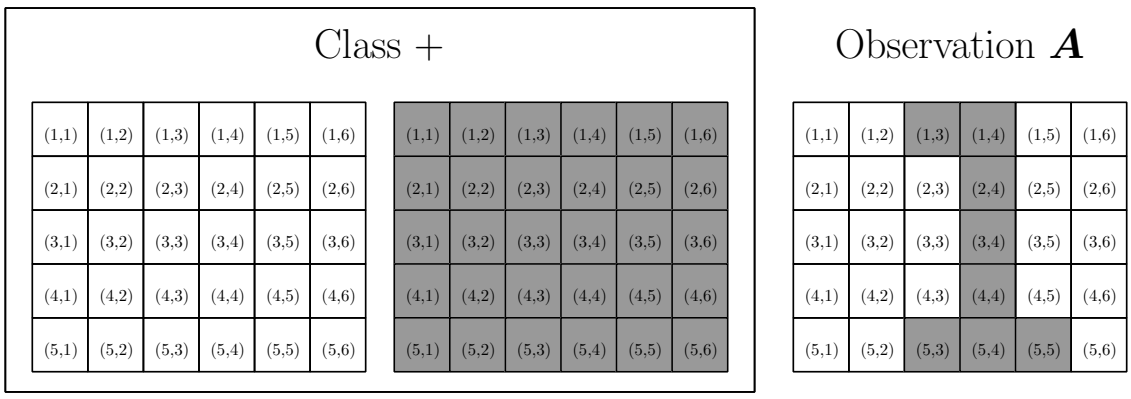


Figure 2.7: An example of deceptive data with respect to the setup of β.

In real applications, the observations of a class are usually more similar to each other than those of the just presented example, although there exist differences between themselves: peculiarities, noise etc. Moreover, a training sample is usually

bigger than that of the example, so that it covers a great diversity of these singularities. Therefore, during training some addresses occur more frequently, for a greater number of observations than others. If one of the addresses which represent an observation being classified also represents most of the observations of a given class, this could be considered a strong evidence that this observation belongs to that class. However, if this coincidence regards a single observation of that class, it might be just an accident.

The original WiSARD model does not consider these details, what makes it subject to saturation: the situation when discriminators can match observations excessively dissimilar to those used for their training. The most common effect of saturation is the increased number of draws regarding the identification of the best matching class during classification. Of the approaches for this problem analyzed during this research, most of them rely on counting the addresses frequencies (BLEDSOE e BISSON, 1962; GRIECO *et al.*, 2010), although there exists alternatives which are based on modifications of the training algorithm (BRADSHAW e ALEKSANDER, 1996; TARLING e ROHWER, 1993). It was previously remarked (CARVALHO *et al.*, 2013; LUDERMIR *et al.*, 1999) that these methods indeed improve WiSARD accuracy, but at the expense of a higher computational cost.

2.3 Open Set Recognition

From the previously provided definition of classification, it can be noticed that an enormous variety of situations in human life can be characterized as a task of this sort. Despite some similarities, a detailed analysis of these situations may lead to distinct classification variants, which requires specific theoretical treatment, or at least benefits from it. The fundamental classification problem regards two classes, and assumes the prior availability of a data sample which reflects most of the characteristics of the population under consideration. From this landmark, numerous tasks are derived.

One of these derived tasks is the multi-class classification, which requires the identification of the best matching class among a collection of more than two classes. WiSARD can tackle problems of this kind directly, as it can maintain knowledge regarding all known classes and calculate how well they match an observation to be classified just like in the two-classes case. Classifiers like the Multilayer Perceptron need to be adapted to handle multiple classes: suppose there are k classes; it is possible to consider $k(k-1)/2$ binary classification subtasks (one for each pair of classes); an alternative approach defines k subtasks, so that in each of these one of the classes is considered positive while all the others are regarded as negative; at

last, the results of each subtask are aggregated in a global classification verdict.

Another variant is the one-class (or unary) classification (KHAN e MADDEN, 2009). Solutions to problems of this kind should classify observations as elements of a reference class or outliers to it. The reference class is well-represented in the training sample, while negative examples may be available or not. If the training sample contains negative examples, it is improper to tackle a one-class problem as a default binary classification task: the expectation that negative data provides an overview of the negative class is valid for the binary case, but not for the unary case; therefore, the classification should always be done based on the reference class, but negative training data can be used to adjust the decision criterion. The most common use of unary classification is for *anomaly detection*, a task focused on recalling abnormal data.

These alternatives to the default version of classification regard very clear changes in the base: for multi-class classification, the number of classes to be modeled is increased; for one-class classification, the same number is diminished. The number of classes is one of the most basic and important characteristics of a classification problem, but the analysis of more subtle variations of other aspects is also valuable.

2.3.1 Closed Set Assumption

A premise of classification is the presence of representatives of all possible classes in the training sample, what is called the *closed set assumption*. As the name implies, this is not necessary for open set recognition: beyond known classes, there could be an even larger collection of unknown classes whose observations should be identified as so. The difference between a classification and a recognition task relies on which are the possible answers to a request for class prediction: a classifier always outputs its best guess for the true class of an input observation; for recognition, if none of the known classes appears to be the true class, the response is to consider the observation an outlier to all known classes. The action of ruling an observation as an outlier, which occurs in detection and recognition tasks, is referred to as *rejection*.

Unfortunately, a great number of works which ignore the necessity of rejection can be found in the literature. These works proposed solutions to problems which are regarded as regular classification tasks, although dealing with data from non-targeted classes is not only theoretically possible but expected in practice. This could lead to poor results when a solution of these is used in a real-world application, out of its testbed. Such questionable modeling can be found in various contexts: fault detection (MIROWSKI e LECUN, 2012) and human activity recognition (ANGUITA *et al.*, 2013) are some examples. Table 2.3 summarizes the differences between open set recognition and its closest relatives.

Task	Target	Training Data	Predictions
Classification	Discrimination between classes	Abundant data of all classes	Label of a known class
Anomaly Detection	Recall of abnormal data	Abundant normal data; few or none outliers	Outlier: yes or no
Open Set Recognition	Identification of data from targeted classes	Abundant targeted data; few or none non-targeted	Label of a target class or 'unknown'

Table 2.3: Differences between open set recognition and related problems.

2.3.2 Openness

An interesting aspect of a task which requires rejection is how important this action is for its accomplishment. This comes from the fact that for different problems, the amount of data which should be rejected may differ: for example, rejection is less useful for the recognition of chickens and ducks among farm animals than among birds in general, as the last group is broader than the first. From this intuition, the *openness* of a given problem is an estimate of the indispensability of rejection for its proper solution. SCHEIRER *et al.* (2013) defined this measure as shown in expression (2.5), using three quantities: C_e, the number of existing classes, which could have to be handled while performing predictions; C_t, the number of classes with observations in the training sample ($C_t \leq C_e$); and C_r, the number of classes which should be later recognized ($C_r \leq C_t$, as the training sample may contain strictly negative examples, which do not prompt the modeling of their classes).

$$\text{Openness} = 1 - \sqrt{\frac{2\ C_t}{C_r + C_e}} \quad . \tag{2.5}$$

A classification problem is "closed" ($C_t = C_r = C_e \Rightarrow$ openness $= 0\%$), what is consistent with the fact that a classifier does not need to reject data. The openness of a detection problem whose training sample contains just positive observations ($C_t = C_r = 1$) grows with the number of all known and unknown classes, C_e. As the number of classes available during training (C_t) increases, the openness diminishes. The opposite happens for the number of target classes, C_r. Those who defined this measure claim that its value always conveniently between 0% and 100%, but this is not true: for example, if $C_r = 1$, openness is negative if more than half of all classes are represented in the training sample (i.e., $C_t > C_e/2$); generalizing this principle, openness is negative whenever $2C_t > C_r + C_e$, an ordinary condition.

2.3.3 From Classification to Open Set Recognition

Open set recognition requires learning not only the differences between target classes but also what differs data of these classes to extraneous data. It is reasonable to consider the adaptation of existing classifiers to this last requirement, as the first is already covered by their functioning: attached to each class prediction it could be provided some sort of confidence rate of such inference. Ideally, this adaptation allows the use of the possibly complex decision criterion established by the classifier in this alternative application. Moreover, the additional computational cost can be reduced by such integration.

A margin classifier, as the Multilayer Perceptron, receives such denomination for the way it operates: it defines a function $f : \mathbb{R}^d \to \mathbb{R},\ \mathbf{x} \mapsto y$ which provides class predictions $\hat{y} = sgn(f(\mathbf{x}))$; consequently, it also defines a hypersurface $f(\mathbf{x}) = 0$, called decision boundary, which divides the feature space delimiting two regions related to each class. For any $\mathbf{x} \in \mathbb{R}^d$, $f(\mathbf{x})$ is nothing but the signed distance between $\mathbf{x}$ and the margin, what naturally induces the idea of using this value to identify outliers: the farther $\mathbf{x}$ is from the margin, the stronger is the evidence that it does not belong to the known classes.

Although this last statement is sensible, to be distant from the decision boundary is a sufficient but not necessary condition to be an outlier. Figure 2.8 illustrates a situation in which a linear-kernel Support Vector Machine, another margin classifier, defined a decision boundary using the observations denoted by circles and squares as the training sample. The distance to the margin of both observations denoted by stars is roughly the same, but the observation closer to the training data is less extraneous than the other one. However, the only information any margin classifier can provide is this observation-to-margin distance. The decision boundary is defined according to the provided data aiming to be a criterion for class prediction as broad as possible: if an observation which is really different from the training sample has to be classified, this should be done as well as possible. This way, the margin which is defined is not optimized for rejection.

As just explained, a confidence rate to be used for rejection is hard to compute for a class prediction realized by a margin classifier. As a matter of fact, this limitation can be related to the kind of probabilistic model a margin classifier is: it tries to approximate $\text{argmax}_y P(y|\mathbf{x})$ using the learned decision function $f(\mathbf{x})$. Classifiers which work directly with the conditional probability $P(y|\mathbf{x})$ are known as discriminative classifiers. On the other hand, generative classifiers estimate the joint probability $P(\mathbf{x}, y)$, from which the conditional probability can be computed. Interestingly, it seems to be acceptable to use the probability $P(\mathbf{x}, y)$ as the desired confidence rate for the association of $\mathbf{x}$ to class y.

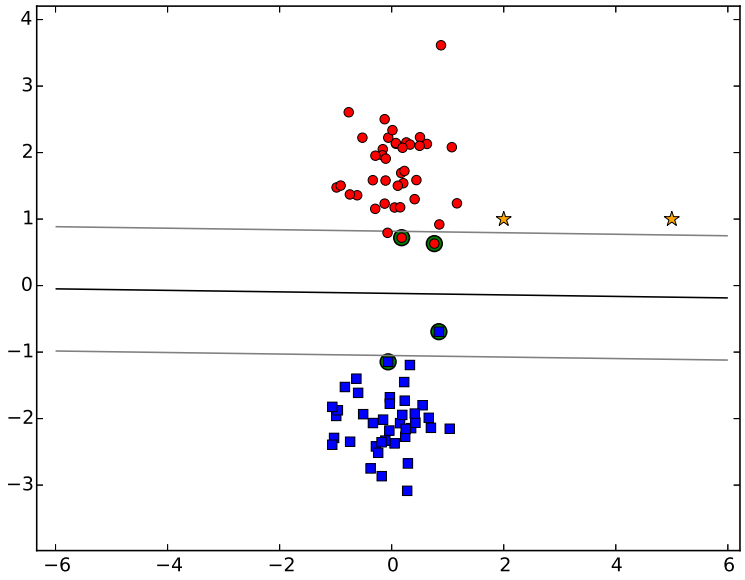


Figure 2.8: Observations represented by stars: equidistant from the margin, but in different conditions regarding being outliers.

However, generative classifiers can not be readily used for open set recognition. That is because they are usually based on probabilistic principles as the Law of Total Probability and Bayes' theorem. The fact that prior probability of the classes is generally unknown in open set problems disallows the use of these tools (SCHEIRER *et al.*, 2014). Besides this, the computation of a good probability density estimation targeting rejection would require a large, noise-free data set (TAX e DUIN, 2008), richer in the informative aspect than a data set to be used just for classification.

There exist approaches for open set recognition in the literature. Many of these are based on discriminative principles: rejection-adapted support vector classifiers (FUMERA e ROLI, 2002; GRANDVALET *et al.*, 2008; JAIN *et al.*, 2014; SCHEIRER *et al.*, 2014; ZHANG e METAXAS, 2006) and ensembles of one-class classifiers based on support vectors (CHEN *et al.*, 2009; HANCZAR e SEBAG, 2014; HOMENDA *et al.*, 2014) are possibly the most common descriptions of methods recently proposed for this task. This can be considered a natural consequence of the popularity of these techniques, previously used in huge variety of closed set tasks. However, for open set recognition, a solution with a generative background could fit in more naturally thanks to its embedded confidence estimation: the adaptation of a solution of this kind looks less painful than the same for a discriminative solution. A promising alternative is the development of a distance-based (TAX e DUIN, 2008) or prototype-based (FISCHER *et al.*, 2015) method, which would have some capabilities similar to generative methods, while avoiding the probabilistic bases which

should not be used in open set tasks.

2.4 Data Stream Clustering

The most basic notion of learning regards the reproduction and generalization of an unknown mapping given some examples in the form of a given input and the desired respective output. Classification is an example of a process of this kind, in which the classes are the elements of the image of such mapping, and this image is entirely known before learning takes place. A classifier works trying to identify features which are alike between observations of the same class and distinct if their classes differ. It is interesting to notice that these same features would still exist for the same data set even if there was no information regarding to which classes the observations in the training sample belong to.

Clustering is the learning activity concerning the discovery of data features which indicate the existence of relations between observations. As a result of this activity, it is possible to define data clusters, grouping observations based on their accordance on the discovered features. The definition of such groups of observations can be interpreted as some sort of "reversed classification": instead of looking for discriminative characteristics of previously defined groups, as a classifier does to class-labeled data, now these characteristics have to be identified so these groups can be defined. This way, the set of features a classifier defines after processing a given training sample is possibly discoverable by clustering the same observations while ignoring to which class they belong to. However, clustering may also lead to other features, which could not be useful for classification, but provide interesting insights of the data under consideration.

Still comparing classification and clustering, an interesting aspect is how these processes are guided while being performed. The performance of a classifier can be unequivocally computed and adjusted according to previously known examples: if the class predicted for some observation is not the expected, this is a unquestionable mistake; a hypothetical perfect classifier would be able to reasonably avoid all mistakes. On the other hand, to estimate how good is the outcome of a clustering procedure is more complicated, as it does not exist a standard criterion for such evaluation. In other words, a clustering problem can have more than one solution, and the best among those can vary according to the chosen interpretive point of view. For these reasons, classification is considered a supervised learning activity, while clustering is labeled as unsupervised learning.

2.4.1 Learning from Data Streams

A premise of most machine learning tasks is the existence of a data set from which the knowledge should be extracted. In this default scheme, learning happens once only, as a step of system functioning whose start and finish are clearly defined. Alternatively, a data stream can replace a data set as knowledge source, serving as an unbounded information provider of the population it regards. In this case, the way learning is carried on needs to be adapted, as it can no longer be realized as a single step. The least which can be done is to incrementally expand and refine the obtained knowledge as new observations become known. An even more challenging use of data streams considers that the attributes of the underling population are not static, so that the more recent observations are more important for its characterization. Therefore, not only learning should be performed incrementally but outdated information should be discarded as well. To learn under these last constraints is what is called data stream mining.

The notion of a data stream is less intuitive than that of a data set. Formally, a data stream $S = (s_1, s_2, \dots)$ can be described as a unbounded sequence of elements. Each $s_i = (\mathbf{x}_i, t_i)$ is a pair consisting of a n-dimensional real vector $\mathbf{x}_i \in \mathbb{R}^d$ representing an observation, and the time stamp $t_i \in \mathbb{R}$ of the instant the observation was input to the learning system. It is worth noticing that two elements s_i and s_{i+1} can be enormously apart in the temporal sense, despite the minimal difference of their indexes. Hiatus as well as bursts of observations can happen during stream processing. To consider $t_i = i$ is a commonly used simplifying assumption (JIN *et al.*, 2014; ZLIOBAITE *et al.*, 2014), also considered during this research to make easier the comparison of the results obtained to those of other works. However, as desirable, it was not established any dependence to such stricter condition.

Clustering a data stream requires more than defining reasonable collections of observations, as in the static version of the problem. Incremental learning is realized updating the previously defined groups by the addition of novel observations as they arrive. This can prompt not only the enlargement and density increase of these groups, but also the merging of groups separated until then. The emergence of new groups also needs to be considered in this incremental functioning. The disposal of obsolete information works the opposite way, shrinking and reducing the density of groups as well as prompting splits and vanishings. The continuous combination of information addition and removal results in the displacement of the maintained groups in the feature space. All this should happen still taking into account the temporal information which is attached to each observation.

2.4.2 Modeling Options

The just mentioned characteristics, common to most data stream clustering tasks, still allow different implementations of some of its parts. One of these parts regards the definition of data obsolescence, and an aging model is responsible for such definition. Any of these models is described through two characteristics: the range of the data stream it encompasses and the weight each covered observation has on current knowledge. For example, the sliding window model (DATAR e MOTWANI, 2016) always considers the ω most recent observations equally important to knowledge definition (ω is a model parameter). Therefore, whenever a novel stream element s_i is received, it should be used to update current clustering definition, becoming as important in this regard as s_{i-1}, while the influence of $s_{i-\omega}$ should be canceled. Another option is the damped window model (CAO *et al.*, 2006), which covers all stream items previously seen, assigning weights proportionally to how recent the inputs are, according to a decay factor.

The landmark window (ZHU e SHASHA, 2002) works incrementally adding novel inputs to its range until it is detected a concept drift: a change in the properties of the underlying population, what makes previously seen data less useful for the definition of up-to-date knowledge. When this happens, the window is restarted, so that the information maintained until then is discarded and redefined from scratch, as the data window grows. Handling concept drift in a smother manner, the adaptive window model (BIFET e GAVALDÀ, 2007) expands its range using new observations until change is noticed. This triggers the gradual reduction of window size until the covered data, the most recent items of the stream, are considered stable, drift-free. These two methods assign equal weights to all covered inputs. They also rely heavily on effective drift detection, what can be integrated to the model or performed separately.

Beyond the previously presented formal characterization of a data stream, a practical aspect of their use is the high input rate exhibited by some streams. This shows that, with respect to efficiency, handling data streams is more challenging than the same for data sets: each input has to be processed without considering previous inputs individually, in an unrestricted manner; they can be arbitrarily numerous, even in recent history. Such condition can be translated into this constraint: the stream item s_i is available just until the item s_{i+1} becomes known. The adequate strategy to this setting is to use each item to update a data summary maintained by the learning system. The entire system functioning depends on this summary depth, upkeep cost and support to incremental learning and forgetting.

Any implementation of this data summary is valid, as long as it properly supports learning under the aforementioned conditions. Despite this freedom of choice,

possibly all existing alternatives for this matter were derived from the same basic idea: to maintain a collection of micro-clusters, structures which represent tiny groups of observations which are similar to each other in the spatio-temporal sense. These micro-clusters fill the gap between raw input, the stream items, and high-level system output, a description of the present data clusters: they compose an intermediate layer which fulfils the efficiency requirements of stream processing while maintaining enough information to suitably feed the definition of the true clusters; a direct connection between these ends would probably fail in both aspects.

This micro-clustering strategy establishes two separate procedures in system operation (SILVA *et al.*, 2013): the data abstraction (or online) step, regarding the continuous stream processing and consequent update of the micro-clusters; and the clustering (or offline) step, regarding the description of the current clusters. The clustering step is performed on demand, triggered by some external interaction or automatically, for example, when concept drift is detected (WAN *et al.*, 2009). As this step does not handle the stream directly, it does not have the efficiency constraints of the data abstraction step. Consequently, this step is commonly implemented as a regular clustering algorithm whose input data are the micro-clusters. On the other hand, the data abstraction step can be briefly described as a loop wherein it is updated the micro-cluster to which the most recent observation best fits. Algorithm 2.4 presents the general form of such procedure.

1: **for all** $\mathbf{x}_i$, the observations from the stream **do**
2: Discard information obtained from outdated observations
3: Discard micro-clusters which are no longer useful
4: Find the micro-cluster mc_j which better encloses $\mathbf{x}_i$
5: **if** mc_i is close enough to $\mathbf{x}_i$ **then**
6: Update mc_j definition using $\mathbf{x}_i$
7: **else**
8: Start a new micro-cluster based on $\mathbf{x}_i$

Algorithm 2.4: A generic data abstraction procedure.

In literature (BARDDAL *et al.*, 2015a; BHATNAGAR *et al.*, 2014; CARDOSO *et al.*, 2012; KRANEN *et al.*, 2011) the micro-clusters were described using different structures (e.g., clustering features, grid cells), with their own attribute sets and organization schemes (e.g., tree, linked list, graph). Despite such variety, all alternatives have the same pre-clustering purpose. But they still differ between each other in some aspects: for example, how the best fitting micro-cluster is identified, or the conditions which prompt the creation or elimination of a micro-cluster.

As just stated, the processing of each stream item results in the update of a single micro-cluster. By default, this one-to-one relation is also maintained in the clustering

step: each micro-cluster is used in the composition of a single top-level cluster. In other words, the clusters represent a partition of the observations. This standard is known as *hard clustering*. Its alternative is called *soft clustering* (WAN *et al.*, 2008), wherein each observation belongs, in different degrees, to all clusters. It can be noticed that this partitioning scheme has a closer resemblance to a classification setup, which requires the assignment of each observation to a single class, ignoring possible intersections between classes.

2.4.3 Relation with Open Set Recognition

The described routine of a system for data stream clustering is an instance of the concept of prequential learning (DAWID, 1984): the inputs become known sequentially; for each incoming observation, a decision is made for it; then, the same observation is used to update the decision criterion of the learning system. Regarding data stream clustering, the decision is the identification of the micro-cluster which better encompasses the observation. And the decision criterion update happens when the chosen micro-cluster incorporates the characteristics of the observation: it consequently becomes more apt to absorb other observations similar to the current one in the future; at the same time, the opposite goes for dissimilar observations.

It is also possible that no existing micro-cluster is considered an appropriate match for the observation in question. Such event can have various explanations: the observation is just noise, not being useful for knowledge update; or the formation of a novel cluster in an unpopulated region of the feature space just started. Taking into account this 'no match found' outcome, this micro-cluster querying is similar to the decision taking performed for open set recognition: the micro-clusters play the role of the modeled classes, and a rejection happens when none of them is accepted for absorbing the observation. In some sense, the substitution of a data set by a data stream make clustering more similar to open set recognition. However, the first activity requires knowledge update and disposal of outdated information, has harder efficiency constraints and is unsupervised.

BENDALE e BOULT (2015) proposed recently a solution for a learning task called 'open world recognition', which is similar to open set recognition, but considers the addition of new classes after initial training based on external support. Such activity resembles data stream clustering, but lacks some of its requirements: once a class is added to the knowledge base, its definition is not updated when an observation is ruled as an element of that class; no time factor is considered, so that knowledge can be incremented but it is never forgotten; there is no restriction regarding the availability of past observations after novel ones become known. A data stream clustering system can be easily adapted for open world recognition: for

example, it is enough to use active learning to assign classes to the micro-clusters maintained during stream processing (IENCO *et al.*, 2013).

Chapter 3

WiSARD for Open Set Recognition

Recapping what was previously stated in this text, classification is an activity which naturally models numerous everyday situations. The formal description of default classification considers the existence of just two priorly known classes. The most natural variant is multi-class classification, in which the number of classes is greater than two. Another popular derived task is the identification of elements of a single well-known class among a collection of observations, known as one-class classification, or anomaly detection. As it can be perceived, all these alternatives differ by the number of classes the learning system has to model.

Another task based on classification is open set recognition, in which observations of some classes should be recognized accordingly while inputs which do not belong to the known classes should be rejected, that is, ruled as outliers. To use a classifier for open set recognition, it is necessary to make it capable of identifying extraneous data. Discriminative classifiers, which work based on the conditional probability $P(y|\mathbf{x})$, can only provide the distance between a given observation $\mathbf{x}$ and the decision margin defined during training. This information is somewhat poor for the purpose of rejection. Generative classifiers seem to be naturally more appropriate to the situation in question: the joint probability $P(\mathbf{x}, y)$ they model could be readily used evaluate the pertinence of $\mathbf{x}$ to y. However, the probabilistic foundation of these models does not comprise the reduced notion of the prior probabilities of the classes.

A WiSARD classifier is composed by a collection of discriminators, which are used to evaluate how well an observation fits the classes they represent. Despite the name of such structures (discriminators), WiSARD exhibits some generative capabilities: for example, it is possible to generate prototypes of the known classes through a procedure called DRASiW (GRIECO *et al.*, 2010), a reference to the reversal of the ordinary system operation. To produce such prototypes is possible thanks to how learning is realized by this model, explicitly collecting pieces of infor-

mation from training data for later use. Such generative trait of WiSARD motivated the analysis of its applicability in open set recognition tasks.

As a matter of fact, matching$(\mathbf{x}, y)$ can be interpreted as a proximity measurement between observation $\mathbf{x}$ and class y. Such point of view, which was not explored in depth prior to this research, is particularly important here, where to make decisions based on distances is preferred over some probabilistic computations, as explained in Section 2.3.3. While the notion of distance between objects of the same type, as two observations, is more straightforward, this element-to-set proximity has more facets to be considered. Hence it allows a greater diversity of approaches, and this matching computation can be seen as one of those. A comparison between WiSARD matching and other well-known alternatives is presented ahead in Section 3.1, targeting to highlight some idiosyncrasies of the first.

To perform open set recognition, it is necessary to define a rejection criterion, which regulates the outcome of a classification according to a confidence associated to this action. With respect to WiSARD, this confidence is represented by the estimated observation-to-class proximity. But to learn from available data how to rightfully reject extraneous data requires some further development: how close an observation must be to its predicted class to such classification be considered reasonable? In other words, how to define boundaries which ideally encompass data from target classes only? Section 3.2 is dedicated to these questions.

Section 3.3 presents experiments for practical evaluation of the proposed WiSARD-based system. The procedures which describe these experiments impose the need for rejection in a similar way to what can be encountered in real-world applications. Indeed, some data sets used here were originally used for plain classification, neglecting the intrinsic necessity of handling extraneous data for the problems in question. Therefore, through the experiments not only effectiveness in open set recognition is assessed but the importance of adequate identification and modeling of tasks of this kind is also stressed.

At last, Section 3.4 sums up what was discovered through this research. This includes comments regarding open set recognition as a machine learning task which deserves particular attention despite its relation with classification. It also addresses how the proposed methodology proved to be an useful tool, considering its accuracy and interpretability in the proposed experiments.

3.1 Proximity Measurement

In this section, WiSARD matching is analyzed as a proximity measure. First, the features considered in this analysis are enumerated, together with a brief description of their importance and the subjects they cover. Then, based on the just mentioned

features, an in-depth characterization of WiSARD is given. Next, the considered alternatives for proximity assessment are presented and compared in the proposed terms. At last, the alternatives are compared graphically through various test cases, targeting to provide a better intuition of their characteristics.

3.1.1 Featured Aspects

The first point of view from which the alternatives to be compared are analyzed concerns how the proximity computation procedure is carried on, qualitatively. This perspective prompts various interesting questions. For example, how granular is the output of this procedure? Also, how meaningfully does this procedure provides different answers to distinct queries, or how does it uniquely handles an observation and data set input? And, how sensitive is this procedure to the presence of extraneous elements in the data set? At last, are there parameters which can be used to adjust the execution of such procedure, and how they alter such execution? To observe the available options from this angle can be seen as the most basic and essential analysis to be realized, as it regards their core functionality.

The second aspect which is taken into account is the time complexity of the process, or processes, which are realized in order to obtain a desired proximity measurement. It is interesting to consider separately the cost associated to a preprocessing or training phase and the actual proximity computation, as the context considered here is that of open set recognition, where training is performed once followed by the answering of one or more prediction requests. Depending of the existence of efficiency constraints for the target application, how computationally expensive is proximity evaluation may be considered even more important than the precision of such computation, assuming a trade-off between these points.

The last feature to consider is similar to the previous one: the memory complexity of the proximity measure. During preprocessing/training phase, it is usual to feed some data structure which is later used during prediction phase. If the resources available for task realization are limited, one alternative may be preferred over other one if the last requires the storage of a too large amount of data. Thus, it is interesting to know how much data could be stored at most, and the relation between this quantity and the size of the data set.

3.1.2 WiSARD Matching as Proximity Measure

WiSARD matching is the first option for proximity assessment to be described.

Core Functionality

The granularity of WiSARD matching output is controlled by a model parameter: the number of nodes per discriminator, δ, is the number of distinct proximity degrees which can be assigned. Recalling the examples in Section 2.2.1, if it is assumed that the number of bits which describe an observation is mn, the original WiSARD model requires $\delta \leq mn$, so that for the length of the addresses, $\beta = mn/\delta$, the inequality $\beta \geq 1$ holds. However, it is possible to increase this granularity through *oversampling*, using every bit more than once for addressing, as proposed next.

Let $\mathcal{C}$ be the set of combinations of β elements from a pool of mn bit references, so that $|\mathcal{C}| = \binom{mn}{\beta}$. Any collections of bit references $c_1, \cdots, c_\delta$ used for addressing are nothing but a random sample of $\mathcal{C}$ for which expression (2.3) holds. If this constraint is dismissed, it is established a new condition, $\delta \leq |\mathcal{C}|$, which allows a much higher granularity. Indeed, considering that matching$^*(y, \mathbf{x})$ denotes the matching rate when $\delta = |\mathcal{C}|$ (that is, $\{c_1, \cdots, c_\delta\} = \mathcal{C}$), it can be noticed that matching$(\mathbf{x}, y)$, for any $\delta < |\mathcal{C}|$, is an approximation of matching$^*(\mathbf{x}, y)$, based on a sample of $\mathcal{C}$, so that E[matching$(\mathbf{x}, y)$] = matching$^*(\mathbf{x}, y)$. The bigger δ is, the better such approximation is. Therefore, for a given β, it is not strictly necessary to consider $\delta = mn/\beta$ as stated for the original WiSARD model.

If the observations are originally represented as real vectors, they can be encoded to a WiSARD-friendly format using as many bits as desired, using Algorithm 2.3 or a similar procedure. As the number of bits used to represent the observations grows, a similar effect to that of oversampling can be achieved while still using the original bit sampling policy: the granularity of the output is increased while its stochasticity is reduced. However, oversampling can be used for any binary-represented data, and counters stochasticity more explicitly. Moreover, it reveals an interesting probabilistic aspect of WiSARD functioning. At last, the processing overhead grows with the number of bits used to represent the observations.

Another noteworthy peculiarity of WiSARD matching can be pointed out as follows. First, extending the notation already introduced, consider matching$(\mathbf{x}, X)$ the WiSARD-based matching rate of a set of observations $X = \{\cdots, \mathbf{x}_i, \cdots\}$ and an observation $\mathbf{x}$. Then, expression (3.1) can be used to declare that the matching rate monotonically increases as the referenced data set becomes larger.

$$\text{matching}(\mathbf{x}, X \cup \{\mathbf{x}'\}) \geq \text{matching}(\mathbf{x}, X) \quad . \tag{3.1}$$

This is directly related to the mapping-and-memorization scheme of WiSARD, as well as to the condition of saturation described in Section 2.2.3. Some effects of such monotonicity are positive, while others are not. For example, it is impossible

to move away $\mathbf{x}$ from X (that is, reduce their matching) adding observations to X, what makes such computation "outlier-proof" in this regard. On the other hand,

$$\text{matching}(\mathbf{x}, X) = 0 \;\Rightarrow\; \text{matching}(\mathbf{x}, X \cup \{\mathbf{x}'\}) = \text{matching}(\mathbf{x}, \{\mathbf{x}'\}) \quad ,$$

and this last matching rate can be as big as 1, what discredits WiSARD matching as a global proximity measure given how strong can be the influence of a single observation on its calculations. That is, other alternatives should be preferred when the target application requires something in the sense of an average proximity.

One last detail to highlight is related to nomenclature: WiSARD matching indeed represents proximity, or similarity, and not distance. Measures of both kinds provide information about the difference between locations in a space, but a distance grows with this difference while the opposite happens to a similarity measurement. Moreover, the output range of a similarity measure is a bounded interval (e.g., $[0, 1]$), while a distance is just half-bounded as a non-negative value. As these differences were not relevant to the applications analyzed during this research, in this text the term *proximity* is used to refer to measures of both kinds.

Still in this regard, it is interesting to comment how the length of the addresses, β, alter proximity calculation. While the number of nodes in each discriminator, δ, is related to the granularity of the provided matching rates, β defines what could be understood as the atomic matching amount. The smaller is β, the larger is the region defined from a base data set X that covers observations as $\mathbf{x}$ for which $\text{matching}(X, \mathbf{x}) > 0$. Therefore, while the output of WiSARD matching always lies inside $[0, 1]$, this interval is somehow stretched according to the model parameter β.

Time Complexity

WiSARD has a well-defined training procedure, already presented in Algorithm 2.2. For element-to-set proximity assessment, this can be seen as a preprocessing step. For a training sample X, the time complexity of such procedure is

$$O(|X|\,\delta\,\beta) \quad . \tag{3.2}$$

That is, for each of the $|X|$ observations, δ node updates are realized, and the cost of each of these updates comes down to the definition of a key of β bits. It can be noticed that there is no dependence between such cost and the dimensionality of the observations. However, this dependence is established if data requires to be binarized and a procedure like Algorithm 2.3 is used for this. In turn, the proximity measurement itself, which is carried out as a single matching computation as shown in expression (2.1a), costs $O(\delta\,\beta)$, based on the same reasoning used for analyzing

the training procedure.

Memory Complexity

The discriminator nodes register the occurrence of addresses which are β-bits strings. There are 2^β possible values for these strings. This can be used to asymptotically define the memory complexity of a WiSARD discriminator: $O(\delta\,\beta\,2^\beta)$. This bound can be considered too "pessimist", since it is quite uncommon for a node to register $O(2^\beta)$ addresses: first, such condition presumes that $|X| \geq O(2^\beta)$, while in commonly used WiSARD setups, $|X| \ll 2^\beta$; second, it also presumes that the number of distinct addresses to be obtained from observations in X will be of the order of 2^β, but this is hardly feasible, because in practice observations input to the same discriminator, as examples of a given class, are expected to have addresses in common. An alternative bound is $O(|X|\,\delta\,\beta)$, which can be tighter but still does not take into account addresses coincidences properly.

The expected number of different addresses to be obtained from X can be defined based on a version of the *birthday problem* (SIEGRIST, 1997): assuming an year has d days and birthdays are uniformly distributed, how many distinct dates are expected to be birthdays of p people? Consider $b = (1 - 1/d)^p$ as the probability that a certain date is nobody's birthday. Therefore, the expected number of distinct dates that are birthdays is $d(1 - b)$. Back to WiSARD, $d = 2^\beta$ and $p = |X|$. Consequently, the memory complexity is of the order of

$$\delta\,\beta\,2^\beta \left(1 - \left(1 - \frac{1}{2^\beta}\right)^{|X|}\right) \quad .$$

3.1.3 Comparison to Alternatives in Literature

Mahalanobis Distance

As previously stated, the proximity between two observations is a clearer concept than the proximity between an observation and a data sample. An strategy to perform computations of this last type is to define a sample representative in the form of a virtual observation, and then use it for computations of the first type. The centroid (i.e., the arithmetic mean position) of the sample is the most straightforward candidate for representative, as it is a reasonable summary of the sample. Moreover, the distance to the centroid is closely related to the average distance to each element of the sample:

$$\mathbf{x} - \bar{X} = \mathbf{x} - \left(\frac{1}{|X|}\sum_{\mathbf{x}'\in X}\mathbf{x}'\right) = \frac{1}{|X|}\left(|X|\mathbf{x} - \sum_{\mathbf{x}'\in X}\mathbf{x}'\right) = \frac{1}{|X|}\sum_{\mathbf{x}'\in X}\mathbf{x} - \mathbf{x}' \quad .$$

Compared to WiSARD matching, this distance-to-centroid alternative can be considered superior with respect to both time and memory complexity, for its constant computational cost (besides the $O(|X|)$ cost for centroid definition) and also constant storage space requirement. Its output has a higher granularity than that of WiSARD matching, since it is a "strictly" real value rather than a rate of an integer value. As the influence of a single observation of the sample on centroid definition is inversely proportional to the size of the sample, so it is the sensitivity to outliers, what can be considered just slightly inferior to WiSARD performance in this sense.

One bad aspect of this centroid method is its insensitivity to variations of the data set, as long as these variations are symmetrical with respect to the centroid: $X' = X \cup \{\bar{X}-\mathbf{x}, \bar{X}+\mathbf{x}\} \Rightarrow \bar{X} = \bar{X}'$. This way, observations can be indiscriminately added to the base data sample without affecting proximity computation. Another question is the lack of *locality*, as the proximity between an observation and sample (i.e., its centroid) is minimally related to the proximity to neighboring elements of the sample, what is the exact opposite of WiSARD behavior in this regard.

Mahalanobis distance (XIANG *et al.*, 2008) takes into account not only the sample mean but also how its varies in across different dimensions: with the centroid as origin, consider all possible coordinate axes; the sample observations can be spread in some axes while condensed in others. To depart from the centroid over an axis of the first kind is to move away from data in a slower pace than to do the same using an axis of the second kind. This notion can be used to provide some additional context to the default distance-to-centroid measurement. For a covariance matrix $\mathbf{S}$ of the data sample, the Mahalanobis distance is defined as

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \bar{X})^T \, \mathbf{S}^{-1} \, (\mathbf{x} - \bar{X})} \quad .$$

The need to define the inverse of the covariance matrix is one of the challenges associated to this method, as it may not exist, above all on high-dimensional situations. Approximations of this inverse are valid alternatives. With respect to locality and outliers, Mahalanobis distance is similar in this sense to the centroid-only method. The time and memory complexities of both methods are also similar, differing basically by the extra preprocessing cost of defining $\mathbf{S}^{-1}$. The value of $D_M(\mathbf{x})$ is as granular as possible, and is more sensitive to changes in the base data sample, thanks to the extra information the covariance matrix provides.

Nearest Neighbors

To evaluate the proximity of an observation to a data sample based on which items of the last are the closest to the first is a viable alternative. The nearest neighbors method works based on the identification of these closest in-sample observations,

providing the average distance between these and the query observation. Consequently, this method requires storing the entire data set, what means an $O(|X|)$ memory complexity, at least. For a reduced computational cost, spatial data structures (SAMET, 1995) are used for indexing: some of these structures allow nearest neighbors lookups in $O(\log |X|)$ steps only, while requiring $O(|X| \log |X|)$ steps to be fed. From the memory perspective, WiSARD can be considered superior for its sub-linear complexity on the size of the data sample. Regarding time complexity, none of the two approaches clearly dominates the other.

The locality sense of the centroid method contrasts with that of the nearest neighbors method, which can be considered closer to WiSARD in this regard. As a matter of fact, such behavior depends on the single parameter of this method: k, the number of neighbors to find for each query. If $k = |X|$, the average distance to the neighborhood is simply the average distance to every element in the sample. In most applications, $k \ll |X|$. The setup $k = 1$ is common, despite being the most exposed to outliers. As k grows, the influence of changes in the sample becomes wider, but weaker. Regardless of k, the output granularity is always the highest.

Naive Bayes Modeling

Despite the already described drawbacks related to using conventional probabilistic models for open set recognition, for the sake of completeness one of these models is included in this comparison. A Naive Bayes model (LOWD e DOMINGOS, 2005) can provide an estimate of the likelihood of an event based on past events: here, the query observation is the target event and the data sample contains the past events. Such estimation is based on how likely is the value of each component of $\mathbf{x}$, independently of the others: $p(\mathbf{x}) = \Pi\ p(x_i)$. The value $p(x_i)$ is defined according to statistics over the i-th component of sample observations.

The granularity of $p(\mathbf{x})$ depends on how each observation component is modeled: for example, a model based on counting frequencies of the values of each component can be less granular than a model based on an assumption of the data distribution. However, to assume a data distribution can be considered risky, unless such option is well justified. A decision on this matter also reflects on preprocessing cost, since estimating distribution parameters may be necessary, and diverse methods can be used for this. Ideally, such methods should be based on robust statistics, what guarantees their proper functioning in the presence of extraneous data in the sample. Still about computational cost, for $\mathbf{x} \in \mathbb{R}^d$, time and memory complexity are $O(d)$, what makes this model the best among those considered here in these aspects. At last, how far the query observation is from observations in the sample does not reflects on the measurement the Naive Bayes method provides.

One-class Support Vector Machine

As one of the most used methods for outlier detection, a one-class support vector machine (SCHÖLKOPF *et al.*, 2001) represents another kind of proximity measure, which learns a decision boundary from data through optimization, similarly to a margin classifier during its training. Here, optimization leads to the identification of support vectors, the most marginal observations in the base sample. These support vectors are used to define the limits of the area wherein data is considered regular. Model configuration affects the number of support vectors which are chosen and how bound to the vectors is the margin. Among the considered proximity measures, including WiSARD, none besides one-class support vector machine was designed as a rejection tool.

The time complexity of this method is twofold: training cost is up to $O(|X|^3)$, what possibly disallows handling even moderate-sized data sets; one the other hand, computing the distance to the margin takes $O(v)$ steps, where v is the number of support vectors, and $v \ll |X|$ usually holds. Memory complexity is also $O(v)$. The granularity of a measurement is not only the highest possible, but, as a signed number, it also provides an insight of the pertinence of the query observation to the sample. The support vectors are responsible for the locality aspect of the method, since observation-to-sample proximity can be related to the proximity to those that are most external elements of the sample, which establish the decision boundary. The one-class support vector machine is insensitive to changes in the data sample which do not alter the selected support vectors.

3.1.4 Graphical Analysis

This subsection is devoted to analyze the performance of WiSARD as a proximity measure in some toy examples. The previously described alternatives to WiSARD are used to provide baseline results. The comparison presented in the preceding subsection, Like the comparison presented in the preceding subsection, this practical comparison the observation-to-sample proximity assessment capabilities, as the definition-oriented comparison of the preceding subsection. This practical comparison is still focused on the observation-to-sample proximity assessment capabilities, as the definition-oriented comparison of the preceding subsection.

Each test case follows the same routine: given a base data sample of 100 two-dimensional observations and a delimited area in the space, estimate the proximity between each point in this area and the sample. This proximity was indeed computed for a grid of 10,000 (100×100) points uniformly distributed inside the considered region, a sufficiently big set for a good overview of how differently each proximity alternative works in the same circumstances. The measurements were scaled in

order to indicate the proximity to the sample, as values from 0 to 1, the farthest to closest, respectively. Using these proximity rates, a contour plot was drawn to highlight subareas in which the assessed proximity is similar. A dotted line was used to delimit where proximity rate is greater than zero.

The experiments were implemented in Python. The scipy (OLIPHANT, 2007) and scikit-learn (PEDREGOSA *et al.*, 2011) modules provided the implementations of the considered alternatives to WiSARD: the Mahalanobis distance; the nearest neighbors method, with $k = 5$; a Naive Bayes system, which assumes that values in each of the two dimensions follow a Gaussian distribution; and a one-class support vector machine, with $nu = 0.1$ and $\gamma = 200$. Besides this description, default parameter settings were used. The WiSARD instance uses $\beta = 20, \delta = 200$ and $\gamma = 400$. The previously introduced idea of oversampling was used: each observation $\mathbf{x} \in \mathbb{R}^d$ was represented using $\gamma \times d = 800$ bits; from these bits, $\delta = 200$ addresses of $\beta = 20$ bits were generated; this way, each input bit was used $\delta \times \beta / (\gamma \times d) = 5$ times.

The first test case employs a data sample randomly generated from a Gaussian distribution. Figure 3.1 shows the respective results. At first sight, the most evident difference between WiSARD matching and its alternatives is the irregularity of the provided contour levels. Microscopically, this is an undeniable fact, which is related to WiSARD lower granularity compared to the other methods. However, in a broader sense, WiSARD reflects data idiosyncrasies arguably better than all, thanks to its distinct knowledge matching principle. Such mechanism is inherently discontinuous, contrasting with the smoothness most alternative methods target.

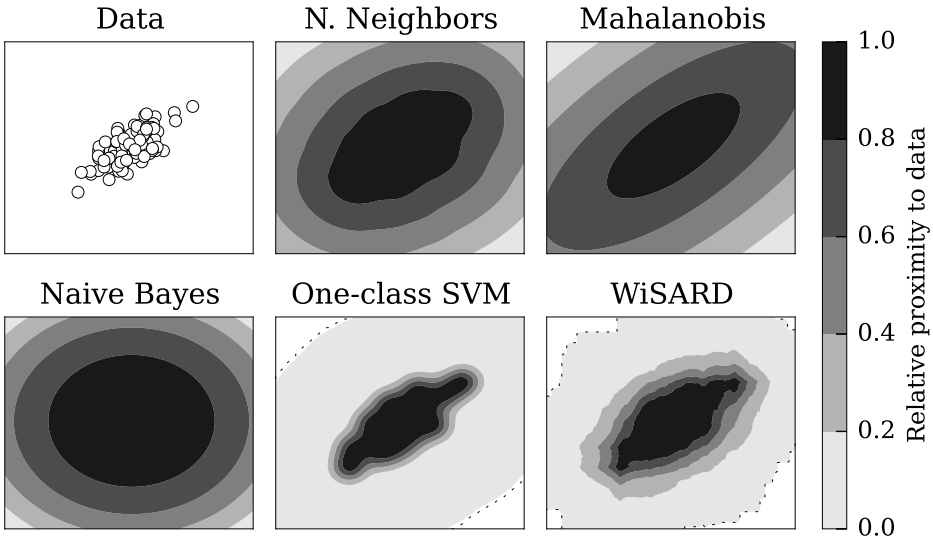


Figure 3.1: The 'Gaussian blob' toy example.

In the second test case a toy data set consisting of two concentric circles of

observations was used. The results are presented in Fig. 3.2. Again, WiSARD roughness is clear, but also its overall proper data representation, clearly superior to those of Mahalanobis and Naive Bayes options. An adequate approach for this test should have an improved sense of locality and enough precision to separate both circles, what was successfully accomplished by WiSARD. Comparing all approaches which could handle the task: the nearest neighbors method concentrated most of its measurements in the interval $[0.6, 1.0]$; on the opposite side, the measurements from the one-class support vector machine are mostly under 0.2, while the range $[0.2, 0.8]$ is underused; WiSARD seems to distribute the measurements more evenly, providing an alternative, possibly more meaningful, proximity assessment.

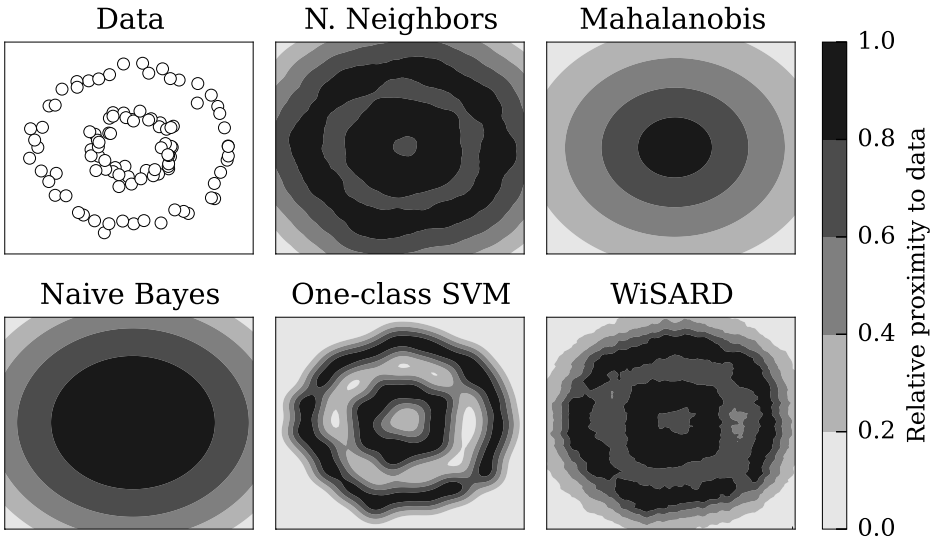


Figure 3.2: The 'two circles' toy example.

Figure 3.3 brings the results of the third test case, in which a data set popularly known as 'two moons' was used. This test confirms what the first two tests show: assuming the independence between observation attributes, the Naive Bayes method is unable to measure proximity to data; Mahalanobis distance (and presumably any method based on global average distance or centroid) lacks locality, what leads to poor results when handling complex, detail-rich data sets; the nearest neighbors method provides adequate results, but is unable to highlight minutiae of the sample; as a proximity measure, the one-class support vector machine yields smooth contours and great deal of detail, but concentrates its measurements on the extremes of the scale, either close to 0 or 1; WiSARD's most patent characteristics are its meaningful proximity assessment and precise reproduction of data peculiarities, but also its irregularity.

Now, analyzing the influence of each parameter on WiSARD matching, each parameter is varied while the others remain as originally set: $\beta = 20, \delta = 200$ and

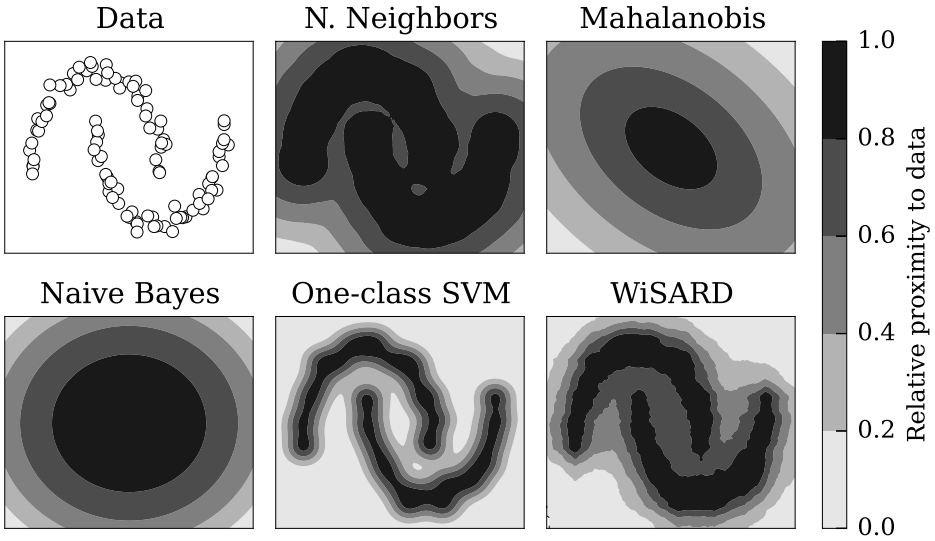


Figure 3.3: The 'two moons' toy example.

$\gamma = 400$. Figure 3.4 shows results for seven different setups of β. As previously stated, this parameter controls matching flexibility, as it indicates how much information (i.e., number of bits) must coincide for the occurrence of a positive feedback from a discriminator node. Therefore, as its value is increased, matching becomes more strict, bound to data sample. From a machine learning perspective, β defines how prone to generalization a WiSARD instance is.

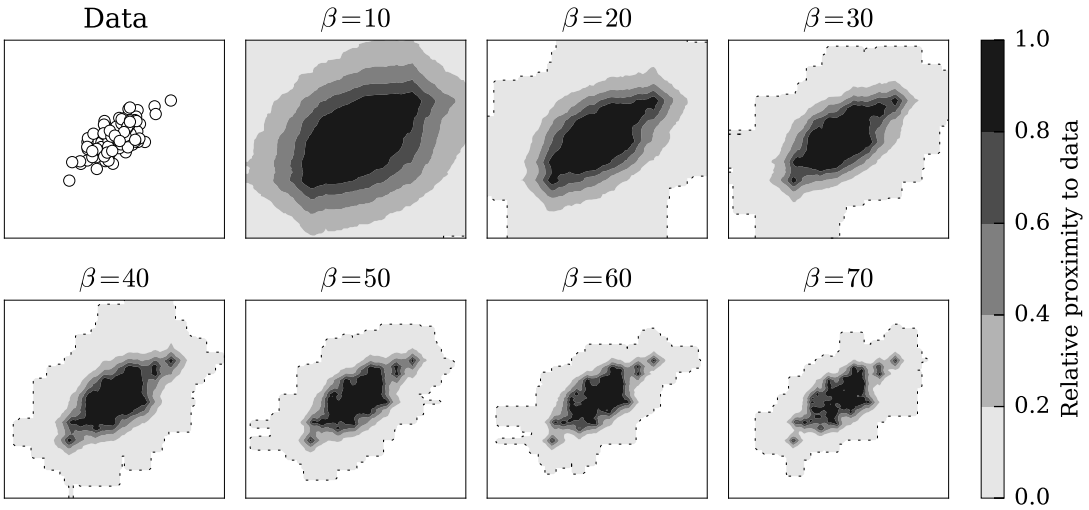


Figure 3.4: Influence of β on WiSARD matching.

The effect of parameter δ setup is illustrated by Fig. 3.5. The contour plots seem to converge to a definitive image following δ increase. Moreover, smoothness is improved during this process. The first proposition is in accordance with the probabilistic interpretation of WiSARD functioning: a larger δ leads to a better approximation of matching$^*(\mathbf{x}, y)$, as well as a more deterministic behavior from the

system. The second proposition is related to granularity: as matching($\mathbf{x}, y$) gets less coarse, it becomes possible to represent properly gradual departure from data.

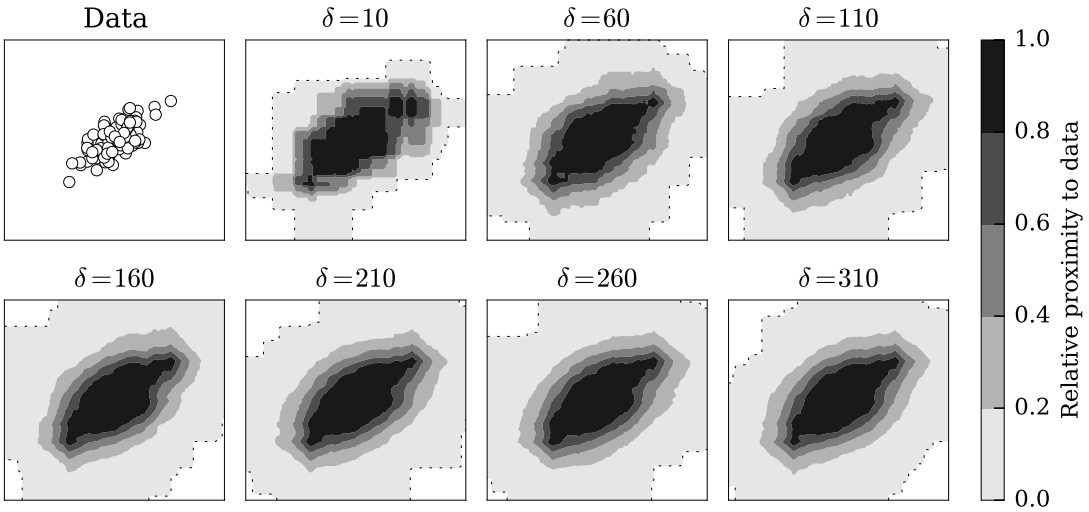


Figure 3.5: Influence of δ on WiSARD matching.

Parameter γ, which defines the number of bits of the binary representations of numeric values, affects WiSARD functioning in a very particular way. As shown in Fig. 3.6, setting this parameter to an excessively low value (in this case, 16 or 32) harms data representability, leading to a poor reproduction of its characteristics. On the other hand, there is no need to set this parameter to an unnecessarily high value (512 or 1024), leading to additional overhead without clear performance improvement. Ideally, γ should be small as possible while still allowing an adequate characterization of the sample.

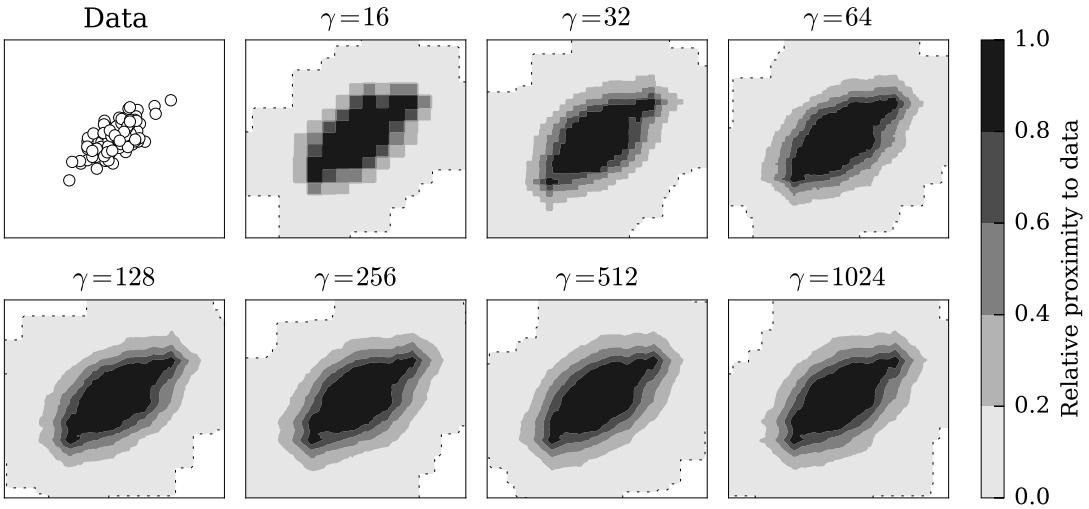


Figure 3.6: Influence of γ on WiSARD matching.

3.2 Computation of Rejection Thresholds

After verifying the validity of matching computation as an observation-to-data proximity measure, it is possible to move on to the next step in the development of a rejection-capable WiSARD. As originally defined, classification comes down to the identification of the best matching class, based on the knowledge kept by its respective discriminator. Therefore, the matching rates of the classes were used just to separate each other in the feature space. Now, considering the general proximity information these measurements provide, it is acknowledged that their use can be extended, for example, to the identification of extraneous data.

One mechanism to label possible outliers is to consider as so any observation $\mathbf{x}$ for which $\text{matching}(\mathbf{x}, \hat{y}) < \alpha$, where $\alpha \in [0, 1]$ is an additional parameter which controls how prone to rejection is the system (CHOW, 1970). This would work if the distributions of matching scores of all classes were the same. However, these distributions generally differ, according to characteristics of training data respective to each class, as sample size, density and homogeneity. Thus, a scheme using individual thresholds $a_{\dot{y}}$ for each targeted class $\dot{y}$ is preferred, allowing to handle unbalanced and noisy data sets properly (FUMERA *et al.*, 2000). Expression (3.3) is the rejection-capable alternative to expression (2.1b) which represents such scheme. The ultimate target is to learn these thresholds from data, making their definition as flexible as possible.

$$\hat{y} = \begin{cases} y' & \text{if } \{y' = \underset{\dot{y}}{\text{argmax}}\ \text{matching}(\mathbf{x}, \dot{y})\} \wedge \{\text{matching}(\mathbf{x}, y') > a_{y'}\} \\ \text{`unknown'} & \text{otherwise} \end{cases} \tag{3.3}$$

The multiple-threshold rejection scheme proposed here was developed in two steps. First, it was analyzed how to efficiently infer some knowledge about the matching of a class and its own elements, according to available training data. From such analysis, it was derived one rejection mechanism which resembles the aforementioned naive alternative, but provides thresholds adapted to each class. The next step comes down to identifying, for each class, the threshold which maximizes a measure of classification effectiveness defined according to a model parameter. These optimal thresholds, whose definition is based on the information obtained in the first step, are employed by a second rejection method also introduced here. Sections 3.2.1 and 3.2.2 are dedicated to each of these parts.

3.2.1 Manual Thresholding

Consider, for a certain class $\dot{y}$ and some training data, that matching$(\mathbf{x}, \dot{y})$ is a random variable, since it depends on $\mathbf{x}$, another random variable. Suppose that although the distribution of matching$(\mathbf{x}, \dot{y})$ is not fully determined, the minimum value this variable can assume for observations whose true class is $\dot{y}$ is known. The intuition behind the rejection method presented next is to use such value for thresholding:

$$a_{\dot{y}} = \min_{\mathbf{x}: f(\mathbf{x})=\dot{y}} \text{matching}(\mathbf{x}, \dot{y}) \quad .$$

In practice, this minimum is indeterminable: the set $\{\mathbf{x} : f(\mathbf{x}) = \dot{y}\}$ is impossible to realize without complete knowledge of data from class $\dot{y}$. However, it could be estimated from the training sample: denoting by $X_{\dot{y}}$ the subset of observations in the training sample which belong to class $\dot{y}$, it could be computed

$$a_{\dot{y}} = \min_{\mathbf{x} \in X_{\dot{y}}} \text{matching}(\mathbf{x}, X_{\dot{y}} \setminus \{\mathbf{x}\}) \quad . \tag{3.4}$$

Naively, this calculation requires performing regular WiSARD training $|X_{\dot{y}}|$ times, as a leave-one-out rotation of the data sample. From expression (3.2), this means a $O(|X_{\dot{y}}|^2 \delta \beta)$ time complexity. This quadratic relation to the size of the data set would reduce WiSARD usual applicability for larger data sets. Therefore, it would be interesting to avoid its establishment. This was possible by means of the exploration of some peculiarities of this model.

In order to reduce the computational cost of $a_{\dot{y}}$ calculation, it is proposed a modification of WiSARD training procedure to embed such calculation, avoiding to perform it separately. Expression (3.4) hints to compute the matching of each observation in $X_{\dot{y}}$, one at a time. As a matter of fact, this can be realized collectively, keeping track of addresses obtained from observations in $X_{\dot{y}}$ but not shared between them. This enables to compute expression (3.5a) efficiently, and subsequently to provide a specialized redefinition of matching: expression (3.5b).

$$\text{exclusive}(\mathbf{x}, X_{\dot{y}}) = \{i : \nexists_{\mathbf{x}' \in X_{\dot{y}} \setminus \{\mathbf{x}\}} \text{addressing}_i(\mathbf{x}) = \text{addressing}_i(\mathbf{x}')\} \quad ; \tag{3.5a}$$

$$\text{matching}(\mathbf{x}, X_{\dot{y}} \setminus \{\mathbf{x}\}) = 1 - \frac{1}{\delta} |\text{exclusive}(\mathbf{x}, X_{\dot{y}})| \quad . \tag{3.5b}$$

Algorithm 3.1 describes the modified training procedure of WiSARD. In a comparison to its original version (Algorithm 2.2), there are basically two changes: every time an address is to be written, its 'ownership' status is updated; and after all addresses are written, a loop over all exclusive addresses is used to compute incrementally $|\text{exclusive}(\mathbf{x}_i, X_{y_i})|$ for all observations. The additional operations represent an increase of the computational cost of WiSARD training, but its time complexity

remains $O(|X|\delta\beta)$, what is as good as possible in this case. After this procedure is concluded, $\text{EXCLUSIVE}_i = |\text{exclusive}(\mathbf{x}_i, X_{y_i})|$.

Let OWNER be an empty dictionary
for all pairs $(\mathbf{x}_i, y_i)$, the training sample **do**
    for all addresses a_j in addressing($\mathbf{x}_i$) **do**
        if $a_j \notin \Delta_{y_i,j}$ **then**
            $\Delta_{y_i,j} \leftarrow \Delta_{y_i,j} \cup \{a_j\}$
            $\text{OWNER}_{y_i,j,a_j} \leftarrow i$ ▷ Adding a new dictionary entry
        else
            Remove entry OWNER_{y_i,j,a_j} ▷ Address is not exclusive
Let $\text{EXCLUSIVE} = 0_{|X|}$ be an array of $|X|$ zeros
for all $\langle (y_i, j, a_j)\ ,\ i \rangle$, entries in OWNER **do**
    $\text{EXCLUSIVE}_i \leftarrow \text{EXCLUSIVE}_i + 1$

Algorithm 3.1: WiSARD training procedure, modified to track exclusive addresses.

Expression (3.5b) and, consequently, expression (3.4) can be easily calculated based on array EXCLUSIVE. This leads to a definition of thresholds strongly oriented to avoid mistaken rejections, so that no element of the training sample would be incorrectly ruled as an outlier if it was ignored during training. Such setting is useful, but in some situations mistaken rejections may be preferred to wrong associations of extraneous data to targeted classes: for example, to reject few observations of a targeted class in order to correctly identify a large amount of outliers is generally interesting. Thus, for an adjustable rejection criterion, expression (3.6) was used as an alternative to expression (3.4). P_α denotes the α-th percentile of the considered values, and $\alpha \in (0, 100)$ is a parameter.

$$a_{\dot{y}} = \underset{\mathbf{x} \in X_{\dot{y}}}{P_\alpha}\ \text{matching}(\mathbf{x}, X_{\dot{y}} \setminus \{\mathbf{x}\}) \tag{3.6}$$

The combination of Algorithm 3.1 and expressions (3.5b) and (3.6) provides a rejection criterion based on what can be inferred about a class from its own observations only. This is particularly interesting for situations in which all training data concerns a single, targeted class, as in various unary classification tasks. Even in this scenario it is still possible to use α to control rejection tendency.

3.2.2 Optimal Thresholding

The manual thresholding scheme which was just described defines $a_{\dot{y}}$ using no observation besides those from $X_{\dot{y}}$. However, there is no reason to avoid employing

observations from $X \setminus X_{\dot{y}}$ to establish a rejection criterion if those are available. Moreover, to use data from other classes looks reasonable considering that such data is extraneous to class $\dot{y}$ and should be rejected accordingly. In other words, to reject observations of the targeted classes which would be otherwise misclassified is just another perspective of the same original goal.

Ideally, $a_{\dot{y}}$ would be set so that

$$\forall_{\mathbf{x}} \text{ matching}(\mathbf{x}, \dot{y}) > a_{\dot{y}} \iff f(\mathbf{x}) = \dot{y} \quad .$$

Such condition wherein the rejection threshold establishes a perfect dichotomy of the observations possibly related to class $\dot{y}$ is generally infeasible: it is quite common to have some observations truly related to $\dot{y}$ but with a low matching value, while the opposite happens for some elements of other classes. Therefore, instead of looking for such unrealistic threshold, finding the best value for $a_{\dot{y}}$ according to some measure of classification effectiveness is the alternative to be used. This can be enunciated similarly to an optimization problem:

$$\begin{aligned} \underset{a_{\dot{y}}}{\text{maximize}} \quad & \alpha(\text{LABELS, PREDICTIONS}) \\ \text{subject to} \quad & \text{LABELS}_i = [f(\mathbf{x}_i) = \dot{y}] \\ & \text{PREDICTIONS}_i = [\text{matching}(\mathbf{x}_i, \dot{y}) > a_{\dot{y}}] \end{aligned} \tag{3.7}$$

Expression (3.7) is defined according to the binary classification task of ruling if observations as $\mathbf{x}_i$ are related to class $\dot{y}$ or not. LABELS is an array which represents the ground truth of such task. PREDICTIONS indicates the labels inferred according to matching computation and a given $a_{\dot{y}}$. Here α represents the aforementioned measure of classification effectiveness. Previously (FUMERA *et al.*, 2000) only accuracy was considered to guide thresholds adjustment. However, any method to rate prediction quality can be employed for this: for example, F-measure (GOUTTE e GAUSSIER, 2005).

The idea here is to obtain a reasonable $a_{\dot{y}}$ by solving expression (3.7) just for the training sample. That is, each of the mentioned $\mathbf{x}_i$ is an observation of X which would be classified with respect to $\dot{y}$. Then, the search for the optimal value of $a_{\dot{y}}$ can be limited to all matching$(\mathbf{x}_i, \dot{y})$ values. Again, Algorithm 3.1 is used for training in order to avoid performing explicitly the leave-one-out rotation of the data set. Subsequently, Algorithm 3.2 is carried out to tackle the aforementioned optimization problem. At last, the time complexity of training becomes $O(|X||\dot{Y}|\,\delta\,\beta)$: the number of targeted classes is denoted by $|\dot{Y}|$; the loop starting at Line 4 of Algorithm 3.2, which dominates the computation of $a_{\dot{y}}$, can be performed in $O(|X|\,\delta\,\beta)$ steps.

1: Let $\dot{y}$ be the targeted class whose optimal threshold $a_{\dot{y}}$ is to be computed
2: **for all** $\mathbf{x}_i \in X$ **do**
3: $\quad$ LABELS$_i \leftarrow [f(\mathbf{x}_i) = \dot{y}]$
4: **for all** $t : \exists_{\mathbf{x} \in X}\ \text{matching}(\mathbf{x}, \dot{y}) = t$ **do**
5: $\quad$ **for all** $\mathbf{x}_i \in X$ **do**
6: $\quad\quad$ PREDICTIONS$_i \leftarrow [\text{matching}(\mathbf{x}_i, \dot{y}) > t]$
7: $\quad$ SCORE$_t \leftarrow \alpha$(LABELS, PREDICTIONS)
8: $a_{\dot{y}} \leftarrow \underset{t}{\text{argmax}}\ \text{SCORE}_t$

Algorithm 3.2: Threshold optimization procedure.

As already mentioned, each class-related rejection threshold is defined according to the best solution of a binary classification subtask. Such solution may vary according to which measure α is picked to evaluate classification effectiveness. The choice of α should consider that, for any of these subtasks, class '1' is the targeted class, while class '0' just gathers misclassified observations (i.e., $f(\mathbf{x}_i) \neq \dot{y}$): comparing extreme scenarios, it is better to reject no observation, as the original WiSARD does, than to reject them all, including elements of the targeted classes.

Measures as accuracy are indifferent to distinct roles the classes may have, while others like F-measure are calculated based on a positive (in other words, targeted) class. Consequently, measures of the last kind should be preferred for this use. Still with respect to F-measure, its parameter β can be used to control how prone to rejection is the system: if precision is prioritized, there is a stronger rejective tendency; otherwise, if recall is favored, rejections should occur less frequently. The F_1 score (i.e., $\beta = 1$), which considers precision and recall equally important, was the default standard for threshold optimization used in this research.

3.3 Experimental Evaluation

In this section a collection of learning tasks with open-set premises are described, as well as the results obtained when these were approached with rejection-capable WiSARD-based systems which follow the ideas just detailed. Alternative approaches to these tasks, some of which can be found in the literature, are used to provide baseline results for comparison. Through these experiments it can be noticed how harmful it is to tackle recognition problems with regular classifiers, ignoring the existence of extraneous data. Indeed, some data sets used here were, before this work, only considered for classification. Therefore, the introduction of each data set is followed by an exposition of its open-set nature.

Aiming to provide a rich description of each task, their openness is indicated

together with other relevant information. However, instead of its original formula, expression (2.5), an improved redefinition of this measure, shown in expression (3.8), was used. Mind that C_e (the number of existing classes), C_t (the number of classes with observations in the training sample) and C_r (the number of classes which should be later recognized) have the same meaning in both of them. This revision of openness assessment was motivated by the following reasons: such new version assures that openness $\in [0, 1]$; and it is reasonable to relate a greater number of classes to be recognized to a smaller openness. This second point is consistent with the fact that targeted classes are expected to be comprehensively detailed in the training sample, helping to portrait the task domain more precisely than available data from other classes.

$$\text{Openness} = 1 - \sqrt{\frac{C_r + C_t}{2\ C_e}} \quad . \tag{3.8}$$

As in Section 3.1.4, the experiments were developed in Python, and used the Scikit-learn module (PEDREGOSA *et al.*, 2011) which provides implementations of popular learning algorithms (SVM, Naive Bayes and Nearest Neighbors) and performance metrics (Precision, Recall and F_1 score). The P_I SVM (JAIN *et al.*, 2014) implementation was kindly provided by its authors. The default parameter settings were used unless otherwise specified.

3.3.1 Anomaly Detection

The 'DGA' data set (MIROWSKI e LECUN, 2012) regards power transformers in one of two possible states: operating regularly, as desired, or in the imminence of failure. The challenge here is to rule if a transformer is faulty or normal, according to the concentration of 7 gases dissolved in its insulation oil. This is a small data set, composed of 50 'normal' and 117 'faulty' observations. Originally this data set was used for binary classification. As a reasonable alternative, it was considered here an anomaly detection task, aiming to identify abnormal data.

Regarding this data set, previously reported results were obtained considering random train-test data splits. However, it makes sense to consider the existence of a single normal state, opposed to various abnormal, faulty ones: power transformers can deviate from their standard functioning in many ways. In practice, it is impossible to guarantee that all abnormal conditions are known a priori. An accurate reproduction of the concrete task related to the DGA data set should feature such incompleteness of the training sample. Since plain random partitions of the data set do not ensure such condition, a suitable alternative to those was employed: Algorithm 3.3 describes how train-test splits in the aforementioned mold were generated; in short, instead of single faulty observations, clusters of them were split into the

training and test samples.

1: Let KMeans$(X, n) = \{C_1, \ldots, C_n\}$ be a partition of X in n clusters
2: **function** MakeSplitsDGA(data set X, $s \in \{1, \ldots, 9\}$)
3: SPLITS $= \varnothing$ ▷ SPLITS is a set of train-test splits of X
4: $C \leftarrow$ KMeans$(X_{\text{faulty}}, 10)$
5: Let $SC = \{C \text{ choose } s\}$ be the set of all s-combinations of C
6: **for all** $SC_i \in SC$ **do**
7: $T_{\text{faulty}} \leftarrow \cup_j SC_{ij}$ ▷ SC_i is a set of sets of faulty observations
8: Let T_{normal} be a random 80% excerpt of X_{normal}
9: $T \leftarrow T_{\text{normal}} \cup T_{\text{faulty}}$
10: SPLITS $\leftarrow$ SPLITS $\cup \{(T, X \setminus T)\}$
11: **return** SPLITS

Algorithm 3.3: Generator of train-test splits of the DGA data set.

The openness of the sample partitions provided by function MakeSplitsDGA varies according to its parameter s: if each cluster of faulty observations C_i is considered a class, a lower s means a smaller number of classes in each training set T, as well as more classes in its testing counterpart $X \setminus T$. To assess the influence of openness in this task, different values of s were used: 2, 5 and 8. For each of these three values, MakeSplitsDGA was called 100 times, generating a mass of partitions of the original data set. Additionally, 5000 splits from random 5-fold cross validation settings were also used, for the sake of comparison to a closed-set classification scenario. The reported results regard each train-test split in the 4 groups just described. Table 3.1 summarizes the information about these groups.

Characteristics	Tasks			
	$s = 2$	$s = 5$	$s = 8$	5-fold CV
# Train-test splits	4500	25200	4500	5000
Targeted classes (C_r)	1	1	1	2
Known classes (C_t)	3	6	9	2
Existing classes (C_e)	11	11	11	2
Openness	57%	44%	33%	0%

Table 3.1: Characteristics of tasks based on the DGA data set.

Two tWiSARD ('t' stands for threshold) versions were tested: one using the manual thresholding scheme, with $\alpha = 5$; and another whose thresholds were optimized according to $\alpha = F_1$ score; other parameters of both were set as $\beta = \delta = \gamma = 100$. It is also reported the performance of the following alternatives, with respective parameter setups: a 5 nearest neighbors classifier; a Gaussian Naive Bayes classifier;

a SVM and a 1-vs-all P_I SVM, both with $C = \gamma = 10$; a one-class SVM, with $nu = 0.005$ and $\gamma = 0.025$; a WiSARD, with $\beta = \gamma = 100$ and $\delta = 10$. These settings were obtained in a best-effort search and provided optimal F_1 scores with respect to 'abnormal' class. P_I SVM (JAIN *et al.*, 2014) represents the state of art regarding open set recognition. One-class SVM and tWiSARD with $\alpha = 5$ were trained using data from the 'normal' class only.

Figure 3.7 illustrates the results of this first experiment. It shows four bar groups, related to each task based on the DGA data set. From left to right, the tasks are ordered from the highest to the lowest openness. This way, it is possible to observe some patterns related to such variation. For example, the overall performance grows as openness diminishes, what is expected using richer training data. All regular classifiers (the first four alternatives) obey this trade-off. However, the one-class SVM, a detection-oriented method, best performed in the highest openness scenario. The three rejection-based methods stand out among the rest, producing top results regardless of the openness level.

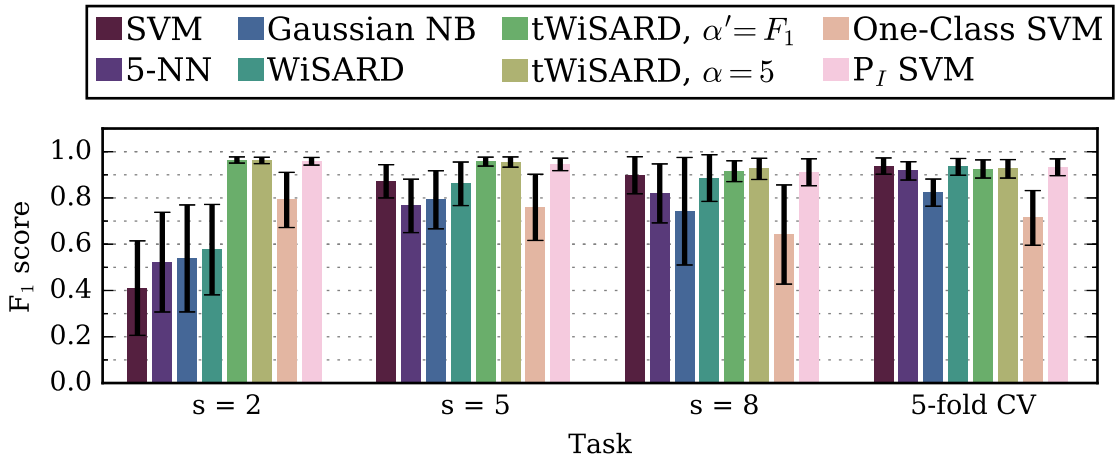


Figure 3.7: Results for the tasks based on the DGA data set. Error bars endpoints are mean ± standard deviation. The reported F_1 scores regard the 'abnormal' class.

Statistically, both tWiSARD versions excel: according to Wilcoxon signed-rank tests with a significance level (α) of 0.01, they were superior to any other tested alternative in all three open-set scenarios. However, in the 5-fold CV setting, SVM, WiSARD and P_I SVM were, by a thin margin, the top performers. Despite this fact, it would be reasonable to choose any of the two tWiSARD alternatives to be used for a recognition task based on the DGA data set wherein the openness level was unknown: on average, they produced the best results of this experiment. At last, in three of the four tasks tWiSARD with $\alpha = F_1$ score performed as well or better than tWiSARD with $\alpha = 5$ for most of the train-test splits.

3.3.2 Multi-class Recognition

It was just shown how a two-class classification task may be better interpreted as an open set recognition problem, with a single targeted class. This is also possible in scenarios with more than two classes, what requires the discrimination between classes of interest as well as the identification of data extraneous to all of them. These two goals are conflicting in some way: observations which would be correctly classified can be mistaken as outliers. Therefore, it is necessary to find an equilibrium to avoid spoiling good class predictions while still rejecting accurately. An interesting question in this regard is: can such balance be found using data from the targeted classes only, without using extraneous data during training? This was analyzed through the experiment described next.

For such purpose, the 'UCI-HAR' data set (ANGUITA *et al.*, 2013) was employed. It is, quoting its authors, "an Activity Recognition database, built from the recordings of 30 subjects doing Activities of Daily Living (ADLs) while carrying a waist-mounted smartphone with embedded inertial sensors". Each observation is a collection of 561 statistics of the sensor readings. However, in this work just a subset of 46 attributes was used: those related to the mean of the readings. This data set is composed of over ten thousand elements, each of them related to one of six activities (i.e., the classes): 'Walking', 'Upstairs', 'Downstairs', 'Sitting', 'Standing' and 'Laying'.

As the DGA data set, the UCI-HAR data set was first used for classification. This way, each of the six classes was represented in both training and test samples. However, in practice, activities beside those known a priori can be realized in an unprecedented way (HU *et al.*, 2013), and they should be recognized as so. In order to mimic a realistic human activity recognition task, in which not all possible activities are known and modelled, each of the six classes was omitted at a time from training: the train-test splits of the data set were defined by a total of 40 5-fold cross-validation runs; each of the 200 test sets was processed six times, considering the same respective training sets, except for the class left out. Thus, in each train-test round, $C_r = C_t = 5$, $C_e = 6$ and, consequently, openness $\approx 8.7\%$.

The same group of methods compared in the anomaly detection experiments is employed here, except for the one-class SVM, which can not handle multiple classes. These methods are enumerated next, with respective parameter setups: a 5 nearest neighbors classifier; a Gaussian Naive Bayes classifier; a WiSARD classifier; two tWiSARD versions, one with $\alpha = 10$ and another with $\alpha = F_{2.5}$ score; a SVM; and a 1-vs-all P_I SVM, with $P = 0.4$; Both SVM and P_I SVM were set with $C = 1000$. WiSARD and both tWiSARD were set with $\beta = 50, \delta = 200$ and $\gamma = 20$.

The UCI-HAR data set features some class imbalance: 18.8% of the data is

related to the most frequent class, while 13.6% belongs to the least frequent one. Despite this difference, all six classes can be considered equally important in the task domain. In order to avoid taking this data set condition into account on the evaluation of the provided predictions, the Macro F_1 score (SOKOLOVA e LAPALME, 2009) was chosen as performance metric for this task. Such choice is explained by the fact this metric is insensitive to class imbalance: the assignment of elements to each class can be seen as a separate binary classification problem, with true and false positives, as well as negatives; the Macro F_1 score is the average of the F_1 scores of these sub-problems.

The results of the experiment with the UCI-HAR data set are portrayed in Fig. 3.8. Each bar group is associated to one collection of train-test rounds in which a class was left out of the training sample. On most cases, the rejection-capable methods had better performances than their regular counterparts: both tWiSARD versions edged the WiSARD classifier on 5 of the 6 tasks, while the same happened for P_I SVM and the regular SVM on the first 4 tasks. For all cases, except for that of class 'Standing', one the last three alternatives was the best performer. These can be seen as evidences which support to take specific care of extraneous data in situations like the one represented by the UCI-HAR data set.

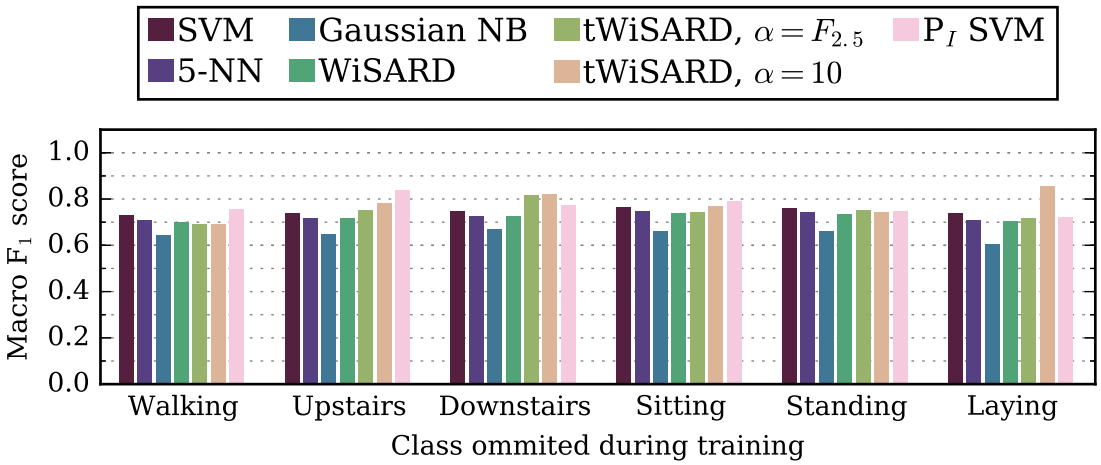


Figure 3.8: Results for the tasks based on the UCI-HAR data set. Error bars were omitted because deviations were negligible.

Wilcoxon signed-rank tests with a significance level of 0.01 support that tWiSARD with $\alpha = 10$ had the best results overall. This can be partially credited to its distinct performance when the 'Laying' class was considered extraneous. The explanation for such outcome is the following: when trying to reject elements of the 'Laying' class, which is the most dissimilar of all, each individual rejection is more likely to be correct; this way, more rejection-prone criteria should perform better in this case. This is confirmed by Table 3.2: when rejecting the 'Laying'

class, tWiSARD with $\alpha = 10$ was the uncontested best alternative regarding not only extraneous-data recall, which grows with rejection tendency, but also precision. This table also shows that on average both tWiSARD versions were superior to P_I SVM rejection-wise.

Omitted Class	Precision			Recall		
	tWiSARD $\alpha = F_{2.5}$	tWiSARD $\alpha = 10$	P_I SVM	tWiSARD $\alpha = F_{2.5}$	tWiSARD $\alpha = 10$	P_I SVM
Walking	0.029	0.185	0.368	0.004	0.111	0.107
Upstairs	0.464	0.459	0.700	0.153	0.479	0.404
Downstairs	0.661	0.518	0.342	0.406	0.672	0.114
Sitting	0.201	0.373	0.388	0.030	0.279	0.125
Standing	0.341	0.288	0.021	0.094	0.173	0.004
Laying	0.464	0.706	0.000	0.062	0.999	0.000
Average	0.360	0.421	0.303	0.125	0.452	0.126

Table 3.2: Rejection performances for tasks based on the UCI-HAR data set.

3.3.3 High Openness

The concept of openness was defined to provide a quantitative degree of complexity of open-set problems, according to the number of classes represented in the training sample compared to those to be handled during the effective use of the consolidated knowledge. The DGA and UCI-HAR data sets, originally considered for classification, were used to define tasks with openness over 57% and 8.7% respectively. This last experiment is an interesting benchmark, designed specifically for open set recognition, with openness over 80%.

The 'LBP88' data set* is composed by elements from two image sets, Caltech 256 (GRIFFIN *et al.*, 2007) and ImageNet (DENG *et al.*, 2009). The first was used to provide train data, while the test sets were composed of positive observations of the first source and negative ones from the last, a cross-data set design which requires the proper rejection of observations from classes not targeted, independently of its origin. In each of 5 rounds, 88 classes were randomly selected. Each of these 88 classes was used once as the one to be recognized, being represented in the training and test samples by 70 and 30 observations, respectively. The remainder of the training sets were 70 (5×14) observations of 5 classes randomly chosen from the 87 negative classes. In turn, the test sets also had 5 observations from each of the 87 classes not targeted. Adding up, the training and test samples had 140 and 465 observations, respectively. Each observation was described by 59 attributes.

*http://www.metarecognition.com/openset/ (accessed 2016/03/06), LBP-like Features, Open Universe of 88 Classes

The open-set nature of the LBP88 data set is quite similar to that of the DGA data set: both are used to define tasks in which one class is well-known a priori and should base the decision criterion, while scarce information from other classes can be used in order to refine such criterion. From another point of view, their respective tasks differ with respect to the desired goal and, consequently, the performance evaluation: for anomaly detection, implied by the DGA data set, the goal is to identify elements extraneous to the base class as abundantly and precisely as possible; for the LBP88 data set, the goal is inversed in a certain way, as the identification of elements of the base class is desired.

The same methods compared through the tasks defined using the DGA data set were reused for the LBP88 data set, but with different parameters: a WiSARD classifier; two tWiSARD varieties, one with $\alpha = 50$ and another with $\alpha = F_{0.4}$ score; a 5 nearest neighbors classifier; a Gaussian Naive Bayes classifier; a SVM, a one-class SVM and a 1-vs-all P_I SVM, all with $\gamma = 35$. WiSARD and both tWiSARD were set with $\beta = 100, \delta = 590$ and $\gamma = 1000$. SVM and its variants were set with $\gamma = 35$. P_I SVM was also set with $P = 0.5$.

Figure 3.9 depict the results for this experiment, described by three different performance measures: recall, precision and F_1 score. These three distinct points of view help to highlight some interesting details. Regular classifiers (the first four alternatives) exhibit a higher recall but also a lower precision level than the rejection-capable methods (the last four alternatives), what leads to worse overall results, represented by the F_1 score. Among this last quartet, P_I SVM had the poorest performance: despite achieving a good recall level, the effect of its relatively low precision on F_1 score is noticeable. This can be compared to tWiSARD with $\alpha = F_{0.4}$, which had the best recall level inside the group just mentioned, but also top results regarding precision and F_1 score.

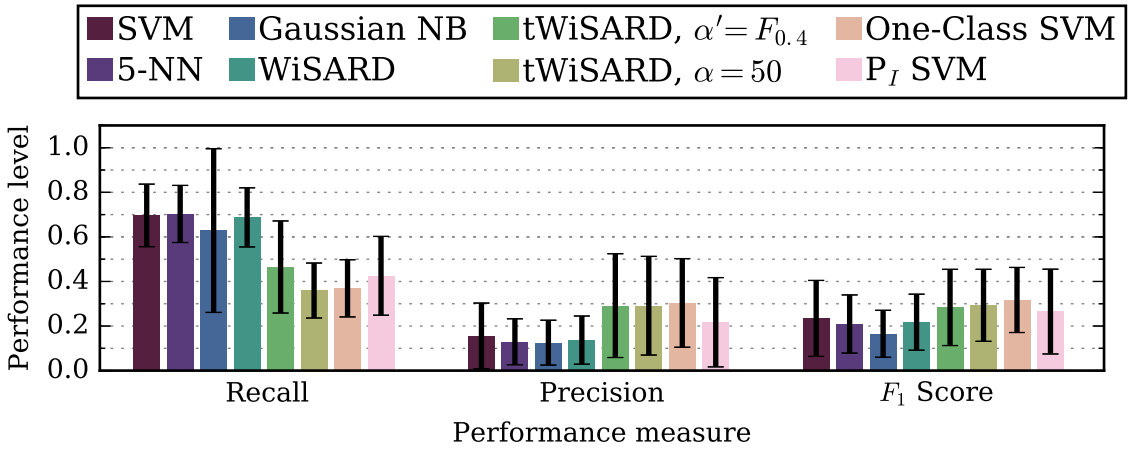


Figure 3.9: Results for the experiment on the LBP88 data set. Error bars endpoints are mean ± standard deviation.

Considering Wilcoxon signed-rank tests with a significance level of 0.01 over the F_1 scores obtained by the tested methods, the one-class SVM was the single top performer. tWiSARD with $\alpha = 50$ and with $\alpha = F_{0.4}$ score had the second and third best results, respectively. However, giving a little priority to recall over precision, tWiSARD with $\alpha = F_{0.4}$ score could be considered the best alternative. Still concerning overall performance, it can be noticed that the two best methods (one-class SVM and tWiSARD with $\alpha = 50$) work using data from the targeted class only. This can be seen as an evidence that available information about extraneous classes may be misleading and produce negative effects on performance. In other words, depending on characteristics of the outliers as variety, frequency and others, it may be wiser, safer, to avoid drawing conclusions based on scarce data about those.

3.4 Concluding Remarks

Classification will always be one of the most basic, ubiquitous and important machine learning tasks. However, despite its value, plain and simple classification should not be used whenever it seems to be possible, just because it is popular, or familiar: to take into account details which would be ignored in a classification setting may lead to an improved, more informative problem definition and, consequently, to better results. Open set recognition is a classification-derived task, which considers the existence of outliers to all classes known a priori, in order to classify unlabeled data more accurately.

Because of its proximity to classification, some approaches to open set recognition found in the literature were built on top of regular classifiers. While this is not wrong, it requires special attention to the differences between these tasks, which should guide the adaptation of those previously existing methods. tWiSARD was developed with such requisite in mind, based on the recognition-friendly WiSARD classification model. Such conception boosts the use of a well-established learning model in situations wherein it is necessary to define more strictly the boundaries inside which it is possible to make conscious decisions. In order to enable this, the provided analysis of WiSARD matching for proximity assessment was indeed useful.

The results of the experiments considered in this research are insightful, highlighting some interesting characteristics of the data which did not emerge during the exclusive use of the classifiers to which the proposed approach was compared. This way, tWiSARD was not only effective combining classification with precise identification of extraneous data. It also provided singular points of view of some data traits, even through the comparison of the performances of its manual-thresholding and optimal-thresholding versions. All these facts can be regarded as evidences in

favor of the applicability of the methodology just introduced.

Chapter 4

WiSARD for Clustering Data Streams

Clustering is a powerful tool to knowledge discovery. It is also versatile, providing valuable information for the analysis of a huge diversity of data sets. However, how data is kept has changed since research on data mining began: the local, static data sets now share space with data streams, unbounded data collections. Data stream clustering (GAMA, 2010) can be accomplished incrementally updating the extracted knowledge as the stream is processed. This incremental learning means not only the expansion of the information already gathered but also the disposal of outdated knowledge. In this context, this means to track the surge as well as the vanishing of clusters, besides changes of characteristics as positioning and shape. Besides those aspects, it is also important to keep in mind that mining data streams impose stricter performance constraints compared to those considered while handling regular data sets.

This text presents henceforth an approach to data stream clustering, based on WiSARD memory-based artificial neural network (ANN) model (ALEKSANDER *et al.*, 1984). This model was chosen as the starting point of this research because: although it was introduced in 1984, recent works in the literature (CARDOSO *et al.*, 2015, 2016a; GRIECO *et al.*, 2010) provided a novel perspective of its features, which could favor the accomplishment of the targeted task; compared to other weightless models, its adaptation to feed from data streams appeared to be less cumbersome, assuring the proper functioning and original properties of the model while supporting incremental and decremental learning from temporal data; at last, the relatively low time complexity of WiSARD, one of its patent characteristics, could help to cope with high input rates during data stream processing, a common setting in real applications.

The fundamental idea of this adaptation of WiSARD to data stream clustering is to use discriminators as micro-cluster representatives. One premise of clustering is

Observation	Class	Best Match	Action	
			Classification	Clustering
$\mathbf{x}_a$	A	A, 78%	$x_a \Rightarrow \Delta_A$	$x_a \Rightarrow \Delta_A$
$\mathbf{x}_b$	A	B, 34%	$x_b \Rightarrow \Delta_A$	$x_b \Rightarrow \Delta_B$
$\mathbf{x}_c$	B	A, 02%	$x_c \Rightarrow \Delta_B$	$x_c \Rightarrow \Delta_{new}$

Table 4.1: An exemplified comparison of WiSARD regular and proposed (i.e., for classification and clustering, respectively) learning process. The discriminator which represents class A is denoted by Δ_A. For classification, any given observation always feeds the discriminator representing its true class. For clustering, since the true class is unknown, the best matching discriminator is chosen, unless its compatibility is considered too low.

that there is no information about the true classes of input observations. WiSARD was conceived for classification, with its discriminators being responsible for modelling the classes to be recognized from their respective training examples. In order to adapt this classifier for clustering, the selection of training examples among unlabeled ones became part of the learning process. This way, any given discriminator would be fed with unlabeled data, but not before checking the compatibility between such data and the pattern represented by the discriminator. Table 4.1 shows how this modification would affect WiSARD operation in some hypothetical cases.

Using this idea as reference, some parts of Algorithm 2.4 would be translated as follows: Line 4 is nothing but a search for the best matching discriminator similar to expression (2.1b); Line 5 is similar to compare a matching rate to a rejection threshold; Line 6 is a regular absorption of $\mathbf{x}_i$ by the discriminator representing mc_j; Line 8 results in the addition of a new discriminator to WiSARD, which absorbs $\mathbf{x}_i$.

This idea is supported by some interesting facts: a discriminator is natively an incremental learner; there is no restriction to adding or removing discriminators, since they exist independently; WiSARD provides a richer feedback than just a distance to a decision boundary as discriminative classifiers, what enables decisions beyond class prediction (CARDOSO *et al.*, 2015); data abstraction units of various approaches to data stream clustering depict data samples based on statistics as mean and variance (AGGARWAL *et al.*, 2003; BARDDAL *et al.*, 2015b, 2016; CAO *et al.*, 2006), which can be considered less informative than the arbitrarily-shaped summary a discriminator can provide (GRIECO *et al.*, 2010).

Despite these appealing premises, in order to provide a complete data stream clusterer, this work successfully overcame limitations and accomplished challenges from past works on the same subject (CARDOSO *et al.*, 2011, 2012, 2016b). One of such accomplishments regards information disposal. Mining data streams requires control over learning as well as unlearning, ignoring past data according to some

criteria. WiSARD training mechanism supports knowledge expansion whenever it is required. However, properly discarding outdated information from its knowledge base was explored for the first time in this research. Section 4.1 shows how the targeted online, continuous learning process was realized.

An issue in WiSARD use is the decrease of its discrimination effectiveness when dealing with unbalanced data collections. This results from how this model works, looking for matching feature values between observations to be recognized and examples of the learned patterns. Therefore, a pattern with more examples has a greater chance to comprise a larger variety of feature values and, consequently, to be considered a good fit to unlabeled data. Since it would be unrealistic to assume that clusters always result from a balanced partition of all data, an adaptation in this sense was necessary. How this was achieved is detailed in Section 4.2.

Section 4.3 provides an overview of the proposed data stream clustering system, aiming to make clear how its components relate to each other and how information is gathered, processed and stored during data stream processing. A conceptual diagram is used in this regard, portraying the whole solution in a high level of abstraction. After all, considering 'the big picture' is indispensable to fully understand how the ideas introduced here work together. An algorithmic description is provided as well, what makes reimplementing the proposed system and reproducing the obtained results easier. Moreover, it is also explained how offline clustering is realized based on knowledge continuously managed during stream processing.

An experimental evaluation of the WiSARD-based clusterer comes in Section 4.4. The results obtained were compared to those of state-of-the-art alternatives, in order to to establish the practical importance of the contributions of this work. All considered methods were tested in a variety of settings, so that their performance could be analyzed when subjected to different learning conditions: batch clustering of regular data sets, incremental clustering of medium-sized data sets, and online clustering of large, streaming data sets, the focus of this research. Both synthetic and real-world data sets were employed, for the analysis of results obtained in a testbed as well as in naturally occurring conditions.

Section 4.5 brings some final comments on the developed approach to data stream clustering. Some general guidelines for the applicability of the proposed method are provided, according to constraints on its operation and characteristics of input data. The achieved contributions, as well as major differences between ideas presented here and related ones found in the literature are also indicated.

4.1 Unlearning and Knowledge Refreshing

The outlook provided so far is very positive with respect to adapting WiSARD to cluster data streams. However, how Lines 2 and 3 of Algorithm 2.4 were implemented based on this artificial neural network model remains unclear. These two lines regard the disposal of outdated information, in two distinct ways: the first is related to cancelling the influence of observations on aggregated knowledge so to keep it up-to-date; the second one concerns the proper ending of the life cycle of micro-clusters when they cease to exist. A description of the approach for each of these follows.

4.1.1 Data Obsolescence

With respect to WiSARD, the influence of an observation on knowledge is materialized by the storage of the addresses obtained from this data point in the nodes of a discriminator. Therefore, the cancellation of such influence comes down to deleting these addresses. In the context of mining data streams, these deletions should happen as past data become obsolete. This can be directly related to the sliding window aging model, which considers data expirations as pinpoint events in the stream timeline. This way, it was assumed that any observation contributes integrally to current knowledge until it expires, when its participation is voided by the removal of its respective addresses. This opposes gradual obsolescence of the damping window aging model, which would require the maintenance of addresses weights, resulting in some additional and undesirable computational workload.

Since the original WiSARD model was not intended to deal with temporal aspects, it had to be modified in order to keep additional information regarding data recency. However, this had to be done without neglecting efficiency, as one of the most basic requirements to work with data streams. Then, to enable the characterization of expired addresses, a time stamp was attached to each entry of the RAM nodes, indicating the last time it was recorded (by Line 6 of Algorithm 2.4). Alone, such change does not alter the computational complexity of model functioning: time- or memory-wise, it represents a cost increase of a constant amount.

Moreover, a process to continuously detect expirations had to be embedded into WiSARD operation. The most naive strategy for this is to check all entries of every node of every discriminator. This memory full scan is as expensive as learning from scratch the entire knowledge base, what is clearly impractical. Such operation is unnecessarily expensive, above all in the context of streams: as data is organized sequentially, just a small portion of it becomes obsolete at one time. Thus, it would be reasonable to consider least recent data as primary candidates for disposal.

Based on such idea, the following policy for time stamps management was defined: a Least Recently Used (LRU) dictionary is used to keep a reference to each

entry of all RAM nodes; every time an address is recorded, its reference is updated as the most recently used. An assumed property of an LRU dictionary is that all its entries are always sorted according to their recency. Therefore, using this structure makes data processing slightly more expensive, since elements of the dictionary need to be reordered every time an new observation arrives. On the other hand, it expressively simplifies the identification and removal of expired addresses: now it is enough to keep checking and deleting the least recently used entry of the dictionary (and its respective RAM node entry) until it is a reference to a up-to-date element.

4.1.2 Micro-Clusters Life Cycle

Summing up what was described so far: when designing the addition of a time dimension to WiSARD, the sliding window aging model showed itself as the best choice, compared to other alternatives, to support the intended system functionality; the characterization of outdated information was performed based on time stamps which indicate the last time each RAM node entry was recorded; references to these entries were kept sorted in a LRU dictionary, targeting to optimize expired addresses identification and disposal. Thus, it was shown how outdated information can be efficiently disposed from all discriminators, so that they serve as up-to-date representatives of data micro-clusters. This regards Line 2 of Algorithm 2.4 only.

Considering this first part of the problem solved, to characterize useless discriminators (Line 3 of Algorithm 2.4) becomes trivial, based on the following reasoning. During stream processing, the knowledge a discriminator possess is expanded when new addresses are added to its RAM nodes, and it is contracted by the removal of expired addresses. Suppose that some time after creation, a discriminator Δ_k becomes "empty" (i.e., $\forall j, \Delta_{k,j} = \varnothing$), because all its addresses expired. Consequently, from then on this discriminator will be unable absorb other observations, since it can not match any of their addresses (i.e., $\forall \mathbf{x}_i, \text{matching}(k, \mathbf{x}_i) = 0$). It can also be said that the knowledge stored by this discriminator was reduced to nothing. Thus, it can be discarded as it is no longer useful.

While the emergence of micro-clusters is explicitly considered during stream processing, prompting the inception of new discriminators, their evolution and disappearance is more transparent. A discriminator is fully active as long as there are addresses related to it, so that it can be chosen as the best fit for incoming observations and, consequently, add new elements to its collection. Meanwhile, addresses which represent outdated information are discarded. When there are no addresses associated to the discriminator which represents a micro-cluster, it vanished. Changes in the collection of addresses of a discriminator represent micro-cluster evolution, moving inside the feature space as well as expanding and shrinking. Thus,

how much of the feature space is covered by a discriminator varies continuously, so that it becomes more or less able to absorb observations over time.

4.2 Cluster Imbalance and Saturation

After making this WiSARD-derived system capable of dealing with temporal aspects, it seemed to be ready to cluster data streams. Unfortunately, this was not true. Because of the way WiSARD learns, memorizing data portions, it is strongly susceptible to have its classification effectiveness harmed when dealing with class imbalance: when at least two of all classes have a significantly different number of examples in the training sample. Despite the absence of information about classes for clustering, some sort of imbalance is still possible and harmful: at some point during stream processing, one discriminator could have a noticeably larger number of observations associated to itself than others; consequently, from then on such discriminator would have the highest chance of being considered the best fit for all incoming observations.

4.2.1 Countering Imbalance with Normalization

A closer inspection at how imbalance affects WiSARD was the first step for the development of a countermeasure for this problem. Suppose that a discriminator Δ_k absorbed a single observation, $\mathbf{x}$. Hence each of its nodes contains just one address: $\forall j, |\Delta_{k,j}| = 1$. Now suppose that it absorbs another observation $\mathbf{x}'$ for which addressing($\mathbf{x}$) = addressing($\mathbf{x}'$). Consequently, although the addresses timestamps will be updated, the set of addresses kept by each node will remain unaltered. From this reasoning, it can be noticed that the number of absorptions a discriminator performed is not necessarily related to class imbalance, opposed to what was previously assumed in the literature (CARDOSO *et al.*, 2012).

It is not hard to notice how the size of the knowledge base of a discriminator alters its chance of absorbing an observation during stream processing. In a hypothetical situation wherein discriminators have a single node (i.e., $\delta = 1$), a discriminator whose node contains the largest number of addresses of all is the most likely to be considered an appropriate match to an incoming observation. The original matching computation of WiSARD, expression (2.1a), does not take the size of the nodes into account. This way, using such expression, the clustering system tends to the undesirable condition wherein there is a single, saturated discriminator covering all feature space, what was experimentally verified.

Still considering $\delta = 1$, matching$(k, \mathbf{x})/|\Delta_{k,1}|$ seems to be a reasonable "normalized" matching rate: this value grows with the matching rate, but shrinks as

$\Delta_{k,1}$ gets larger. To generalize this idea, it is proposed here to define the cardinality of a discriminator according to expression (4.1a), and to define a normalized matching rate according to expression (4.1b). The cardinality of a discriminator is asymptotically equivalent to the area of the feature space itself encompasses: $|\Delta_k| \sim \int \text{matching}(k, \mathbf{x})\, d^n\mathbf{x}$. Moreover, it can be noticed that the denominator of $\text{matching}^*(k, \mathbf{x})$ is the geometric mean of the size of the nodes of Δ_k, what is consistent with the way the sets of addresses are combined for pattern recognition.

$$|\Delta_k| = \left(\prod_j |\Delta_{k,j}| \right) \quad ; \tag{4.1a}$$

$$\text{matching}^*(k, \mathbf{x}) = \frac{\text{matching}(k, \mathbf{x})}{(|\Delta_k|)^{1/\delta}} \quad . \tag{4.1b}$$

4.2.2 Cardinality Weighting

Expression (4.1b) changes WiSARD functioning so that, for a given observation, one discriminator can be considered a better fit than another one, although the last is "closer" to the observation than the first, in the sense of the original matching computation. This can be seen as some kind of penalty for an all-embracing discriminator, whose represented pattern becomes too vague, impossible to define because of how varied its data is. In other words, the answer provided by a more precise discriminator could be preferred, even if it is not the highest rated one.

When considered as a replacement for the original matching rate, its just-introduced normalized version led to better results in experimental tests. This was seen as an evidence in favor of using using such additional information, the cardinality of a discriminator, for refining the matching process for the targeted task. Moreover, there is no additional computational cost as a consequence of modifying WiSARD in this regard: cardinality can be calculated iteratively, alongside expression (2.1a); such calculation requires nothing besides the length of the RAM nodes, what is readily available during system functioning; it is possible to cache a cardinality calculation, since its value remains unaltered until the respective discriminator absorbs an observation or discard some of its content.

Despite these arguments supporting the use of normalized matching rate, to make expression (4.1b) more flexible appeared to be interesting. Such idea resulted from the following: it was noticed that taking feature space coverage into account for pattern recognition can be more or less important, depending on data being processed. After all, WiSARD usefulness is undeniable although cardinality is not considered at all in the original matching calculation. Besides this, imbalance occurrence is not binary, but it can take place at various levels, what was observed in

the data streams used for experimental evaluation.

The outcome of this last reflection is expression (4.2), named adjusted matching rate. Its calculation depends on an additional model parameter μ, the cardinality weight. Setting $\mu = 0$ reduces adjusted matching rate to default matching calculation, while setting $\mu = 1$ leads to the normalized matching rate. As μ grows, the system becomes more prone to choose precise discriminators over generic, all-encompassing ones, despite proximity to data to be absorbed. The adjusted matching rate showed to be the best choice compared to the other two options, allowing the system to be configured to distinct conditions imposed by different data streams.

$$\text{matching}^*_\mu(k, \mathbf{x}) = \frac{\text{matching}(k, \mathbf{x})}{(|\Delta_k|)^{\mu/\delta}} \quad . \tag{4.2}$$

4.3 System Overview

The ideas described in the previous sections are answers to the major issues raised during the development of the intended data stream clusterer. These parts were presented separately, but they integrate with the WiSARD model and, in the end, with each other as well. It is interesting to observe how this system processes and manages information, transforming raw data into high-level knowledge. Figure 4.1 provides such perspective of the whole. It shows how data flows from a source, a generic data stream generation process, towards a sink, a report about data clusters. Some remarks regarding this diagram: it depicts the data stream being continuously generated by some external entity, what contrasts with full prior availability of a regular data set; such external entity also acts as a global time reference.

System offline module works based on a batch agglomerative clustering algorithm, using a collection of micro-clusters representatives, the discriminators, as input to define top-level clusters. In order to make such mechanism operative, it was necessary to define an inter-discriminators similarity measure. Hypothetically, such measure could be the distance between micro-clusters centers estimated according to the knowledge stored in the discriminators. This would be aligned with some works in the literature, which consider micro-clusters hyperspherical entities (AGGARWAL *et al.*, 2003; CAO *et al.*, 2006; KRANEN *et al.*, 2011).

However, discriminators are more flexible, allowing micro-clusters to have arbitrary shapes. Consequently, the distance between estimated centers may not provide a true notion of proximity. To better represent this, expression (4.3) was used to evaluate discriminators similarity. It was inspired in the Jaccard Index (LEVANDOWSKY e WINTER, 1971), but contrasts with it since it is strictly positive: $0 < s(\Delta_a, \Delta_b) \leq 1$. This way, no pair of discriminators is considered completely dissimilar. However, their similarity diminishes as the amount of content not

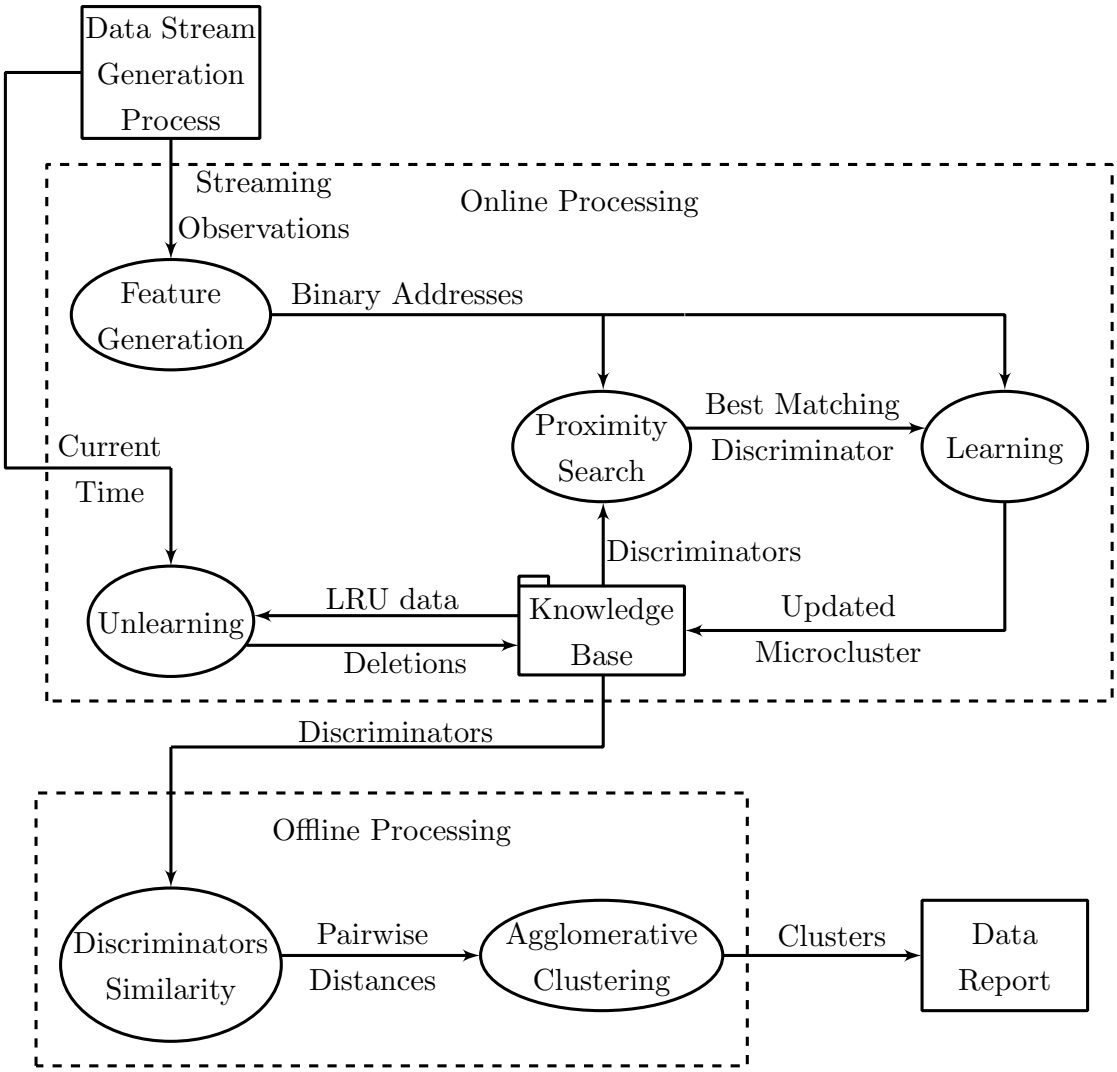


Figure 4.1: A data flow diagram of WiSARD for clustering data streams.

shared between them grows.

$$s(\Delta_a, \Delta_b) = \frac{1 + \sum_i |\Delta_{a,i} \cap \Delta_{b,i}|}{1 + \sum_i |\Delta_{a,i} \cup \Delta_{b,i}|} \quad . \tag{4.3}$$

To conclude the description of the proposed WiSARD-based system to cluster data streams, its online functioning is detailed in Algorithm 4.1, in the same format of Algorithm 2.4. The least recently used dictionary, denoted by LRU, is indexed by triples (k, j, a_j), whose elements regard a discriminator, a RAM node and a RAM address, respectively. Each iteration of the algorithm has a complexity of $O(\delta\beta d)$: the loops starting at lines 2 and 11 require $O(\delta\beta)$ steps; lines 6 and the block starting at line 8 can be realized in $O(1)$ steps; line 7 takes $O(\delta\beta d)$, where d is the number of discriminators which match an observation $\mathbf{x}$ in any degree. The expected value of d varies according to model parameters β and μ.

1: **for all** $(\mathbf{x}, t)$, the streaming observations **do**
2: **while** min LRU $\leq t - \omega$ **do**
3: $k, j, a_j = \underset{k,j,a_j}{\text{argmin}}\ \text{LRU}_{k,j,a_j}$
4: $\Delta_{k,j} \leftarrow \Delta_{k,j} \setminus \{a_j\}$
5: Delete LRU_{k,j,a_j}
6: Delete all Δ_k for which $|\Delta_k| = 0$
7: $k \leftarrow \underset{k}{\text{argmax}}\ \text{matching}^*_\mu(k, \mathbf{x})$
8: **if** $\text{matching}^*_\mu(k, \mathbf{x}) < \epsilon$ **then**
9: Let Δ_{new} be a new discriminator
10: Let k be the index of Δ_{new}
11: **for all** addresses a_j in addressing$(\mathbf{x}_i)$ **do**
12: $\Delta_{k,j} \leftarrow \Delta_{k,j} \cup \{a_j\}$
13: $\text{LRU}_{k,j,a_j} \leftarrow t$

Algorithm 4.1: WiSARD-based data abstraction procedure.

4.4 Experimental Evaluation

The effectiveness of the developed WiSARD-based clusterer was assessed through a collection of experiments. Besides clustering performance, these tests also intended to show how model parameters setup affects its behavior. After all, mining data streams requires finer tuning and better resource management than the same for conventional data sets, what can be directly related to taking into account an additional temporal aspect in order to accomplish automated learning. The experiments were developed in Python, and used the Scikit-learn module (PEDREGOSA *et al.*, 2011) which provides implementations of various clustering algorithms and performance metrics. Moreover, implementations of popular stream-oriented clustering algorithms were used as provided by MOA software bundle (BIFET *et al.*, 2010).

Numerous existing approaches to data stream clustering rely on aging models different of the sliding window model employed in this research. Comparing the results of these approaches to those obtained by the proposed method would be questionable in some sense: if two clusterings are produced using different criteria to decide which are the current observations, they are based on distinct data and, consequently, can not be rightfully compared; the same goes for results obtained considering the sliding window model but with different values for window length, ω.

For a fairer evaluation, conventional methods were used to provide baseline results: K-means (LLOYD, 1982) and average-linkage agglomerative cluster-

ing (DAY e EDELSBRUNNER, 1984), two classical clustering algorithms; and HDBSCAN (CAMPELLO *et al.*, 2015), a state-of-art batch clusterer. This way, to assess the quality of a clustering which is provided at a given time instant t, batch algorithms were run to generate using the current observations (i.e., $\{\mathbf{x}_{t-\omega+1}, \ldots, \mathbf{x}_t\}$) as input. Moreover, online clustering algorithms, DenStream (CAO *et al.*, 2006) and Clustream (AGGARWAL *et al.*, 2003), were adjusted targeting to establish similar criteria to weight stream elements despite the use of different aging models: the larger the sliding window length was set, the smaller was the decay factor considered for the damping window model.

All experiments were performed using data for which label information was available. Such information was not exposed to clustering algorithms during their functioning, but it was used to evaluate the provided data partitions. This same experimental setting was used in numerous recent works on data stream clustering (BARDDAL *et al.*, 2015b, 2016; CARDOSO *et al.*, 2012, 2016b; JIN *et al.*, 2014; WAN *et al.*, 2009). This shows how data stream clustering validation is generally performed using external evaluation measures, which are based on ground truth knowledge regarding data labels. On the other hand, internal evaluation measures, computed according to clusters shape, topology and other geometric properties, are used less frequently (HASSANI e SEIDL, 2016).

The following clustering validation indexes were employed: V-measure (ROSENBERG e HIRSCHBERG, 2007), Adjusted Rand Index (ARI) (WAGNER e WAGNER, 2007), and Adjusted Mutual Information (AMI) (VINH *et al.*, 2010). Except ARI, whose lower bound is -1, the values of all these measures range from 0 to 1, with greater values associated to clusterings similar to ground truth labeling. The use of measures based on combinatorics (ARI) and information theory (the remainder) aim to present different perspectives of the results. Furthermore, the data stream clustering tasks were evaluated comparing the ground truth labeling and clustering of every ω observations.

4.4.1 Batch Clustering of Synthetic Data

The considered methods were compared in three different settings. The first of these regards synthetic, bidimensional data sets, whose number of observations are in the range from over three hundreds to just over three thousands. Also in this setting, data was considered with no time dimension, as in regular, batch clustering. Such conditions combined with the use of prepared data enable to obtain proper initial impressions regarding the alternatives examined. At last, the use of low-dimensional data allow to visualize the resulting partitions in simple scatter plots.

The five data sets used in this first set of experiments come from two publicly

available data sets repositories: Jain, a 'two-moons'-like data set, D31, which features 31 circle-shaped clusters and Aggregation, with 7 clusters in varied shapes, come from a collection found in the Web[*]; Complex8 and Complex9, which feature, respectively, 8 and 9 clusters in varied shapes can also be found in the Internet[†]. Such variety of characteristics poses an interesting test of learning adaptability.

The WiSARD for Clustering Data Streams (WCDS) system was tested using the following parameter setup: $\delta = 200$ and unary enconding with $\gamma = 200$; since no time information was considered, $\omega = \infty$ and $\mu = 0$; the β parameter was adjusted to each data set, and its value is indicated together with the obtained results. The following alternatives, with respective configurations, were also tested: DenStream, with a specific setup for each data set, which provided the best results that could be achieved, while producing the intended number of clusters; Clustream, K-means and average-linkage agglomerative clustering, targeting the true number of clusters of each data set; and HDBSCAN, with a minimum cluster size of 10 elements.

The results of this first task are depicted in Fig. 4.2, which presents the average performance of the tested options in 20 rounds. In each round, order in which the observations are presented to the streaming algorithms was randomly set. This aims to verify if WCDS, DenStream and Clustream can still produce reasonable results when observations, in a relatively small amount, are not favorably sequenced.

For almost all data sets and quality standards, WCDS was the most effective alternative or exhibits a performance close to the best one, according to Wilcoxon signed-rank tests with a significance level (α) of 0.05. Relatively, its worst results regard data set D31, whose circle-shaped clusters were more properly identified by K-means and agglomerative clustering. DenStream and HDBSCAN, which try to detect the number of clusters automatically, performed distinctly from each other: the last was better than the first on most cases, what could be expected when comparing a batch algorithm with a stream-oriented one. Fortunately, WCDS was an exception of such rule in this set of experiments.

Targeting a richer analysis, the visualization of the clusters defined for a given data set could provide an interesting feedback of the algorithms. Figure 4.3 presents such point of view for data set Complex8, which highlights differences between the tested approaches: only WCDS and HDBSCAN correctly identified the 5 horizontally-spread clusters, the other three clusters were mistaken in different degrees by all methods, with HDBSCAN followed by WCDS being the top performers in this regard. This is an evidence of how handling non-convex groups can still be seen as one of the basic and most challenging targets of clustering.

[*] http://cs.joensuu.fi/sipu/datasets/ (accessed 2016/10/10)
[†] http://www2.cs.uh.edu/~ml_kdd/ (accessed 2016/10/10)

4.4.2 Incremental Clustering of Real Data

In the second and third experiment settings, data sets resulting from measurements of real experiments are employed. Many works found in the literature evaluate methods for data stream clustering based on non-stream data, sequenced according to some criteria, or data streams which are mostly stationary (AGGARWAL *et al.*, 2003; CAO *et al.*, 2006; CARDOSO *et al.*, 2012, 2016b; KRANEN *et al.*, 2011). Although results found this way are valid, using true data streams is appealing, leading to conclusions with a greater chance to be confirmed out of the testbed.

A data collection provided by FONOLLOSA *et al.* (2015) contains, paraphrasing its description in a data sets repository[‡], acquired time series from 16 chemical sensors exposed to two gas mixtures (Ethylene and Methane in air, and Ethylene and CO in air) at varying concentration levels. Thus, each of these two sets of stream data have 16 input attributes, which should be used to predict the concentration level of the gases which compose the mixture being considered. Moreover, each data set has over 4 million observations, and feature more than 70 different concentration levels. Figures 4.4a and 4.4b illustrate class dynamics for both sets.

Targeting to reproduce the conditions of a real-world application, to optimize algorithms parameters directly over full data streams was ruled impracticable. Instead, these parameters were tuned based on clustering of data stream samples. This way, this second experiment setting regards clustering sequenced random samples of the data streams. The parameter setups which provided the best results in this case were used later to cluster the data streams in all their extent.

As in the first setting, no data ageing was considered (i.e., $\omega = \infty$). However, as samples of 20 thousand observations were used, linear processing of such data amount is now expected to have a stronger influence on results. Consequently, the cardinality weight parameter μ was exploited targeting to balance observations distribution between micro-clusters during input processing. In a best-effort search, the following parameter settings were defined:

- for WCDS, $\beta = 300$, $\delta = 100$, $\mu = 100$, unary encoding with $\gamma = 200$;
- for DenStream,
  - for Ethylene-CO, $\epsilon = .0039$ and offline multiplier $= 20$;
  - for Ethylene-Methane, $\epsilon = .005$ and offline multiplier $= 16$;
- for Clustream, number of micro-clusters $= 500$;
- for K-means, k was set equal to the total number of classes of each data set;

[‡] http://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+dynamic+gas+mixtures (accessed 2016/10/17)

- for HDBSCAN, minimum cluster size = 10.

Repeating the experimental procedure of the first setting, 20 test rounds were performed for each data stream. In each of these rounds, a sequenced random sample of all data was used. Figure 4.5 presents the average performance of the algorithms being compared. The stream-oriented algorithms, WCDS, DenStream and Clustream, worked as incremental clusterers, while the others processed all observations as a single batch. Again, for almost all cases WCDS performed at least as well as its alternative with the best result, considering Wilcoxon signed-rank tests with a significance level of 0.05. Moreover, it is interesting to notice that WCDS top performance is partially due to taking into account how input observations are ordered: the same results were not observed when the data stream samples were shuffled before processing, what is acceptable for a stream-oriented algorithm.

4.4.3 Data Streams Clustering

After analyzing WCDS performance when clustering regular data sets, which is a more familiar and steady condition, its behavior when dealing with temporal data was examined. Such gradual exploration of system features enables a clearer understanding of functioning as a whole, highlighting the contribution of each of its parts. Thus, it was designed first an application of WCDS to find clusters from relatively small data sets. Then, its cardinality weighting feature was used in order to process longer data sets linearly. Next, the forgetting mechanism was used to reach the ultimate goal of clustering data streams.

As previously stated, the same data and algorithms parameters setups were used in this last test setting. However, the full data streams were considered instead of a sample of these, what means processing over 4 million observations instead of 20 thousand, a significant increase in input size. Algorithms as K-means, agglomerative clustering and HDBSCAN are unable to process such amount of data at once. As an alternative, their results were obtained by dividing the data streams every 20 thousand observations, and processing each of these parts as a batch. On the other hand, WCDS, DenStream and Clustream processed all data sequentially, handling data obsolescence directly.

While the true number of clusters was assumed to be known in the first test setting, to do the same for this last case would be unreasonable: it is impossible to know at every instant during stream processing the true number of partitions to be defined from current data. On the other hand, depending on the application or domain, the total number of labels can be known a priori. This way, the number of labels was used to define the target number of clusters for K-means, agglomerative clustering and Clustream algorithms. DenStream and HDBSCAN defined the

number of clusters according the their configuration. WCDS also defined the number of clusters automatically: its offline operation still came down to performing agglomerative clustering of its micro-clusters; however, instead of targeting a given number of clusters, the stopping criterion for this procedure was defined based on the inconsistency coefficients of the links of its hierarchical clustering tree (ZAHN, 1971).

The stream-oriented algorithms still required the proper configuration of their data obsolescence mechanisms. In this sense, despite implementing different ageing models, both Clustream and WCDS need the definition of a single parameter, the window length, ω, known as horizon in the context of Clustream. In turn, DenStream depends on the definition of its decay factor, λ, whose interpretation is quite distinct from that of the window length. Despite this, to relate these parameters reasonably was still intended. Then, based on experimental tests, the following setting was defined: $\lambda = 100\,\omega^{-1}$. Moreover, it was used $\omega = 20000$, what defines an enough challenging amount of current data to be considered.

Figure 4.6 depicts the average performance of WCDS and its alternatives while clustering data streams. This task can be seen as more difficult than the previous ones considering the poorer overall performance of the clusterers in this setting. Another interesting fact is the low variance of the results: they are practically negligible; this could be related to the absence of factors as input shuffling and random sampling, featured in the first and second experiment settings, respectively; such stability of WCDS despite the random initialization of its addressing is welcome. As in the previous cases, WCDS was the top performer from a statistical point of view, according to Wilcoxon signed-rank tests with $\alpha = 0.05$.

4.5 Concluding Remarks

Clustering data streams is a challenge of major relevance actually, as it combines one of the most basic and important knowledge discovery tools with an ubiquitous data organization scheme. Although there is a rich variety of works on this theme, there is still room for improvements. An achievement of this research was the definition of the concept of cardinality of a discriminator, as well as its usage to establish WiSARD matching functions which counter data imbalance. Without such modification, the base learning model was prone to saturation, an ill condition in which its discriminative power is severely compromised. Despite being a well-known problem in WiSARD use, effective strategies against it still need to be developed. The ideas presented here are intended to be a valuable contribution in this regard.

This research also led to the definition of a continuous learning system, capable of incrementally storing and discarding streaming information. As far as observed,

the existing alternatives to the problem in question discard expired data on an estimate basis. In order to overcome this condition, it was necessary to redefine of one of the most basic building blocks of this area: the micro-cluster. Using WiSARD discriminators to represent micro-clusters supported the development of an alternative mechanism for the disposal of outdated knowledge. That enabled WCDS to provide results comparable to those of a state-of-art batch algorithm while maintaining the processing granularity of a stream-oriented algorithm.

Practical use of WCDS is supported by its top performance while working under distinct conditions, as shown in the performed experiments. Such conditions represent common, real-life application scenarios of cluster analysis. Moreover, using non-synthetic data in this regard is specially interesting, reassuring the claimed model usefulness out of its testbed. At last, to judge the found clusterings according to a variety of intrinsically different quality standards contributes to a fair comparison of all alternatives. Fortunately, WCDS excelled among all its competitors.

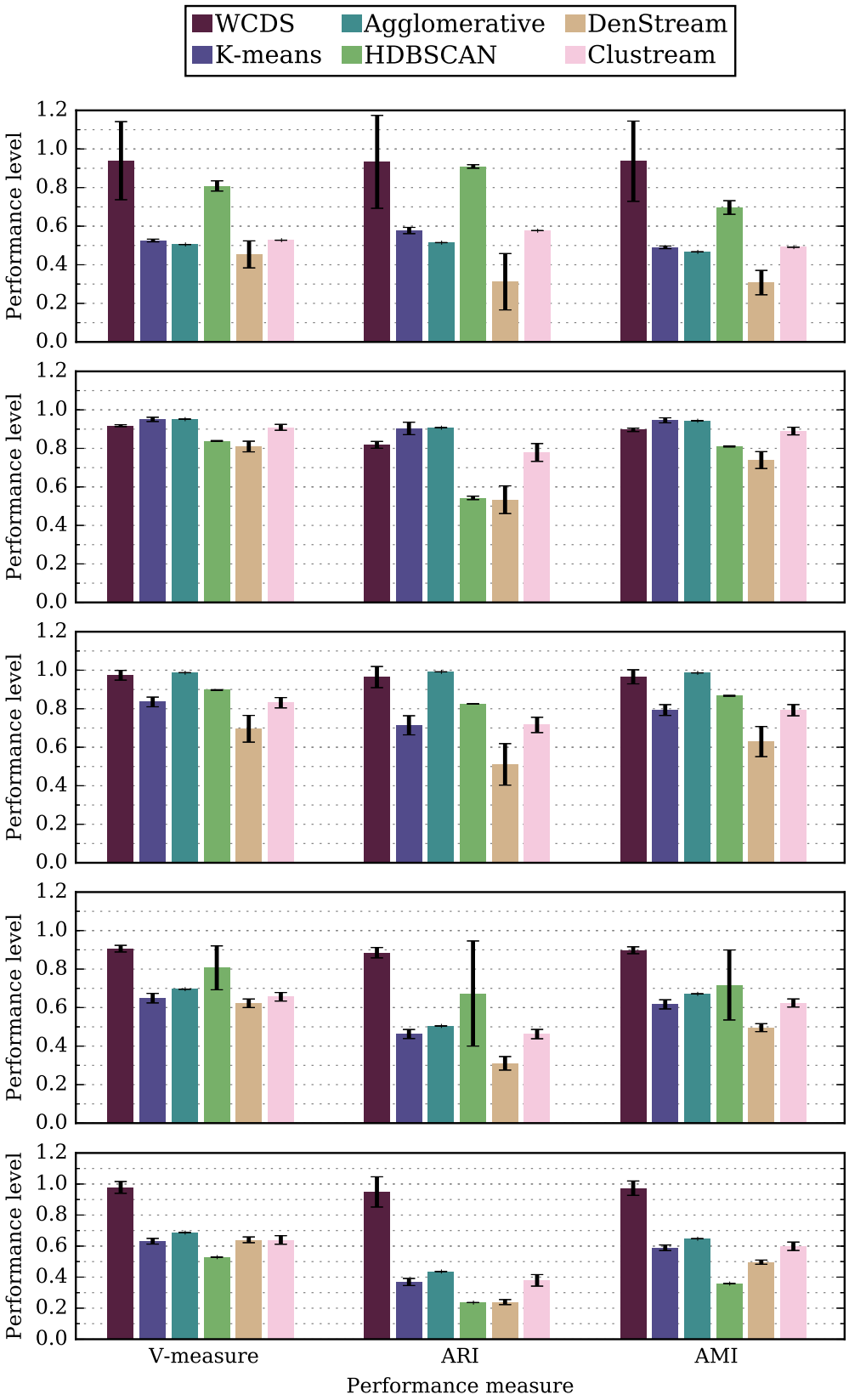
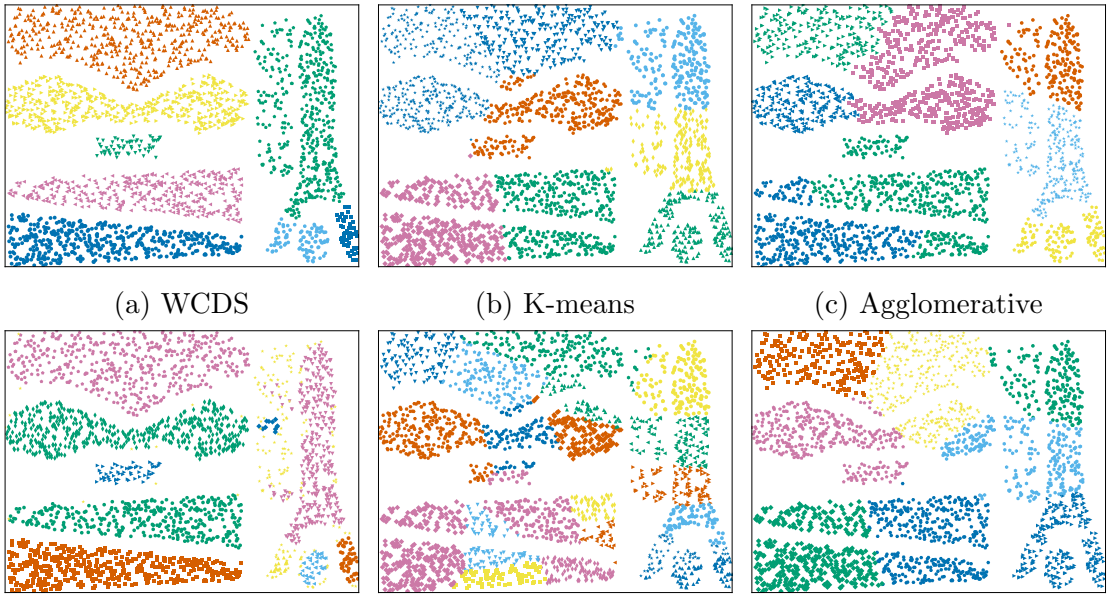


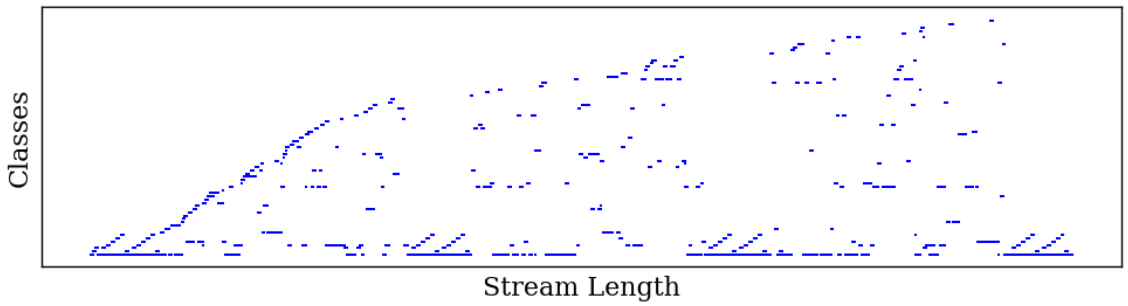
Figure 4.2: Results for the task of batch clustering of synthetic data: bar graphs regarding Jain, D31, Aggregation, Complex8 and Complex9 data sets, respectively. Likewise, for each of these WCDS β was 50, 120, 55, 70 and 80. Error bars endpoints are mean $\pm$ standard deviation.



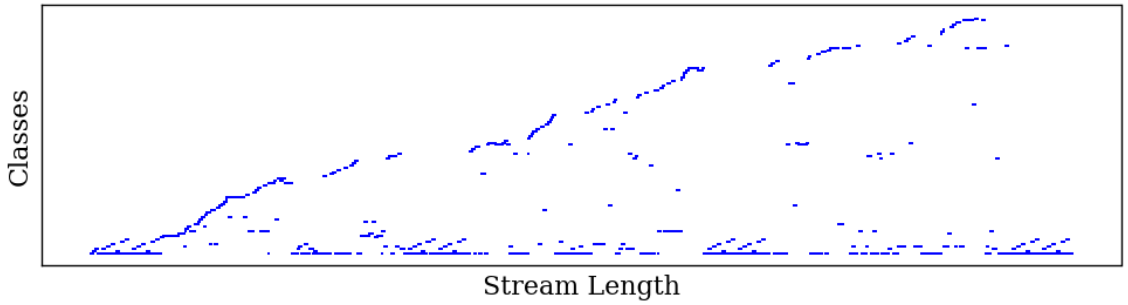
(a) WCDS (b) K-means (c) Agglomerative

(d) HDBSCAN (e) DenStream (f) Clustream

Figure 4.3: Clusters for data set Complex8.

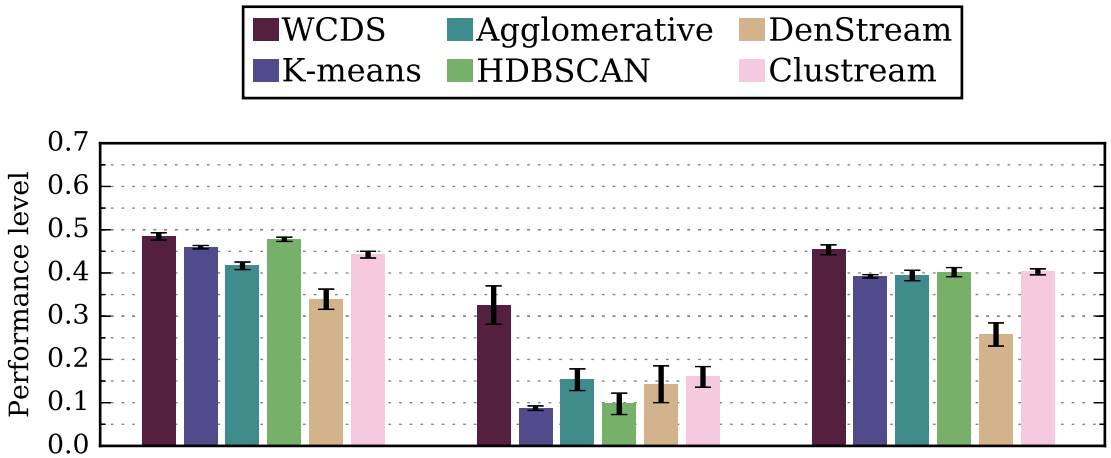


(a) Ethylene-CO data stream

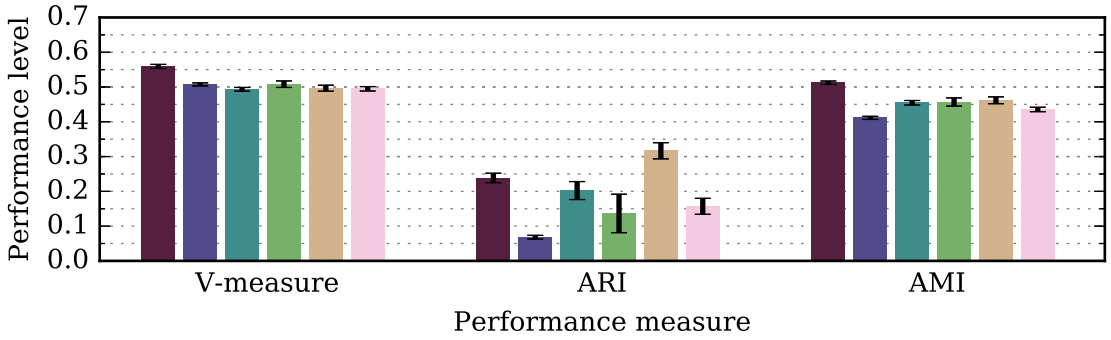


(b) Ethylene-Methane data stream

Figure 4.4: Classes (mixture concentration) variation during stream length.

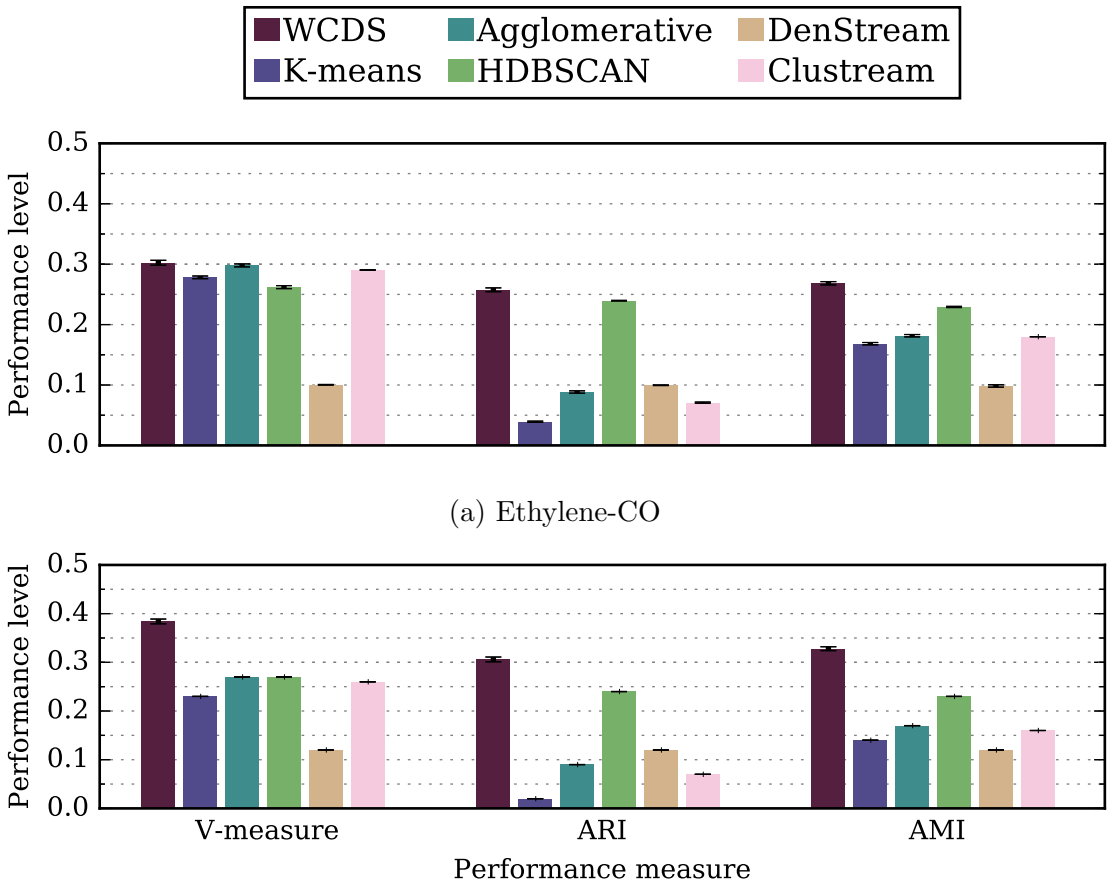


(a) Ethylene-CO



(b) Ethylene-Methane

Figure 4.5: Results for the task of incremental clustering of data stream samples. Error bars endpoints are mean ± standard deviation.



(a) Ethylene-CO

(b) Ethylene-Methane

Figure 4.6: Results for the task of clustering data streams. Error bars endpoints are mean ± standard deviation.

Chapter 5

Conclusion

This chapter completes this research report, pointing the major steps of this work as well as the lessons learned during its accomplishment. After considering the detailed description of these parts provided in the previous chapters, looking at the 'big picture' built from them shall now lead to a different perspective of the whole. This way, no novelty regarding open set recognition, data stream clustering, WiSARD, or any other subject previously addressed in this text, is presented next. Despite such fact, this chapter still provides some fresh information, which could be specially useful to establish links between this and other works. In this regard, some ideas for future works which could be directly related to this one are also given.

This last chapter is organized as follows: Section 5.1 provides an overview of this work, describing its development as well as the reasoning which oriented it; Section 5.2 presents in a succinct fashion the major accomplishments of this work; finally, in Section 5.3 some suggestions for follow-ups of this research are indicated.

5.1 Research Summary

As presented in this text, the research concerned three parts. The first of these could be described as a broad literature review. Initially, such review targeted the identification of the most important concepts and techniques in the field of data stream clustering. However, this also highlighted some common characteristics between such task and open set recognition. To the best of our knowledge, such relation was never considered before, which motivated to expand the original research scope. WiSARD usefulness for data stream clustering was previously considered in the literature, but just superficially (CARDOSO *et al.*, 2011, 2012). This inspired the idea of exploring WiSARD characteristics in both contexts of open set recognition and data stream clustering, targeting to use the know-how obtained in one of them in the other.

Subsequently, a WiSARD-derived approach for open set recognition was developed. Before such feat, it was first analyzed the rejection capability of the base model. Then, the computation of rejection thresholds was not only embedded into model training procedure but also optimized, minimizing the influence of such modification on WiSARD patent speed. The effectiveness and usability of the developed methodology was confirmed by its top performance in various experiments, in which state-of-the-art methods also participated.

The final step of this journey regarded the adaptation of WiSARD to unsupervised online learning, in the sense of clustering data streams. The same core idea of rejection of extraneous data was used. However, additional points had to be addressed: the establishment of a bond between the rejection criteria and clusters evolution; also the disposal of outdated knowledge as more recent observations become available. Solutions to both questions were provided, which enabled to accomplish the development of the intended system. Its experimental comparison to state-of-the-art methods with similar purpose assured its practical value.

5.2 Key Points

This is an overview of the most interesting discoveries of this work:

1. WiSARD matching computation work as a precise, detail-rich, observation-to-sample proximity meter, whose characteristics (e.g., variability, granularity, smoothness) depend on how model parameters are set;

2. exploring the fact that WiSARD functioning does not directly depends on probabilistic principles as the Law of Total Probability and the Bayes' Theorem was crucial for its application in the context of open set recognition, and perhaps could be used for other purposes in the future;

3. compared to regular classification, open set recognition requires additional flexibility because of the openness factor, and the proposed tWiSARD system successfully handled such extra component;

4. open set recognition is related to data stream clustering as some form of rejection is indispensable for both, although only the last requires online learning, up to the level of dealing with the emergence and vanishing of data clusters;

5. to attach to each RAM location a time stamp, instead of a boolean value or a counter as previously done, and to compute the cardinality of each discriminator, what could be done without increasing the computational cost of WiSARD functioning, proved to be enough to power the WCDS system to cluster data streams, despite the apparent simplicity of these ideas.

5.3 Possible Continuations

There are at least two ideas which are directly related to this work and whose development could provide valuable scientific contributions. One of them is a deeper analysis of WiSARD matching as a proximity measure. So far, such perspective was just superficially explored: its utility was experimentally verified, focusing on its application. From a more theoretical point of view, which mathematical properties of such measure can be properly characterized? Moreover, is it possible to formally describe the conditions in which WiSARD matching is equivalent to alternatives like the Mahalanobis distance, Nearest Neighbors, or others?

Another idea is to substitute WCDS fixed-size sliding window aging model by an auto-adjustable one, whose size could vary according to how stable current data is, considering its streaming nature. The LRU dictionary of RAM nodes entries and the cardinality of the discriminators could possibly be used to estimate data stability or the best window length. This way, the disposal of outdated knowledge would become smarter, more dynamic and more flexible, although system structure would remain mostly unaltered. In this case, there is no significant increase of the computational cost while system autonomy is improved, boosting the capability of copping with transient characteristics of the data stream.

Bibliography

AGGARWAL, C. C., HAN, J., WANG, J., et al., 2003, "A Framework for Clustering Evolving Data Streams". In: *VLDB*, pp. 81–92. 4, 4.3, 4.4, 4.4.2

ALEKSANDER, I., THOMAS, W., BOWDEN, P., 1984, "WiSARD, a radical step forward in image recognition", *Sensor Review*, v. 4, n. 3, pp. 120–124. 1, 2.2, 4

ALEKSANDER, I., GREGORIO, M. D., FRANÇA, F. M. G., et al., 2009, "A brief introduction to Weightless Neural Systems". In: *ESANN 2009, 17th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 22-24, 2009, Proceedings*. 2.1.3

ANGUITA, D., GHIO, A., ONETO, L., et al., 2013, "A Public Domain Dataset for Human Activity Recognition using Smartphones". In: *21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013*. 2.3.1, 3.3.2

BARDDAL, J. P., GOMES, H. M., ENEMBRECK, F., 2015a, "SNCStream: a social network-based data stream clustering algorithm". In: Wainwright, R. L., Corchado, J. M., Bechini, A., et al. (Eds.), *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, pp. 935–940. ACM, a. 2.4.2

BARDDAL, J. P., GOMES, H. M., ENEMBRECK, F., 2015b, "A Complex Network-Based Anytime Data Stream Clustering Algorithm". In: Arik, S., Huang, T., Lai, W. K., et al. (Eds.), *Neural Information Processing - 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part I*, v. 9489, *Lecture Notes in Computer Science*, pp. 615–622. Springer, b. 4, 4.4

BARDDAL, J. P., GOMES, H. M., ENEMBRECK, F., et al., 2016, "SNCStream$^+$: Extending a high quality true anytime data stream clustering algorithm", *Inf. Syst.*, v. 62, pp. 60–73. 4, 4.4

BENDALE, A., BOULT, T., 2015, "Towards Open World Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June. 2.4.3

BHATNAGAR, V., KAUR, S., CHAKRAVARTHY, S., 2014, "Clustering data streams using grid-based synopsis", *Knowl. Inf. Syst.*, v. 41, n. 1, pp. 127–152. 2.4.2

BIFET, A., GAVALDÀ, R., 2007, "Learning from Time-Changing Data with Adaptive Windowing". In: *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*, pp. 443–448. SIAM. 2.4.2

BIFET, A., HOLMES, G., KIRKBY, R., et al., 2010, "MOA: Massive Online Analysis", *J. Mach. Learn. Res.*, v. 11 (ago.), pp. 1601–1604. ISSN: 1532-4435. 4.4

BLEDSOE, W. W., BISSON, C. L., 1962, "Improved Memory Matrices for the n-Tuple Pattern Recognition Method", *Electronic Computers, IRE Transactions on*, v. EC-11, n. 3 (June), pp. 414–415. 2.2.3

BRADSHAW, N., ALEKSANDER, I., 1996, "Improving the generalisation of the N-tuple classifier using the effective VC dimension", *Electronics Letters*, v. 32, n. 20 (Sep), pp. 1904–1905. 2.2.3

CAMPELLO, R. J. G. B., MOULAVI, D., ZIMEK, A., et al., 2015, "Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection", *ACM Trans. Knowl. Discov. Data*, v. 10, n. 1 (jul.), pp. 5:1–5:51. ISSN: 1556-4681. 4.4

CAO, F., ESTER, M., QIAN, W., et al., 2006, "Density-Based Clustering over an Evolving Data Stream with Noise". In: *SDM*, pp. 328–339. SIAM. 2.4.2, 4, 4.3, 4.4, 4.4.2

CARDOSO, D. O., LIMA, P. M. V., GREGORIO, M. D., et al., 2011, "Clustering data streams with weightless neural networks". In: *ESANN 2011, 19th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 27-29, 2011, Proceedings*. 4, 5.1

CARDOSO, D. O., GREGORIO, M. D., LIMA, P. M. V., et al., 2012, "A Weightless Neural Network-Based Approach for Stream Data Clustering". In: *Intelligent Data Engineering and Automated Learning - IDEAL 2012 - 13th International Conference, Natal, Brazil, August 29-31, 2012. Proceedings*, pp. 328–335. Springer. 2.4.2, 4, 4.2.1, 4.4, 4.4.2, 5.1

CARDOSO, D. O., FRANÇA, F. M. G., GAMA, J., 2015, "A bounded neural network for open set recognition". In: *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pp. 1–7. IEEE. 4, 4

CARDOSO, D. O., CARVALHO, D. S., ALVES, D. S. F., et al., 2016a, "Financial credit analysis via a clustering weightless neural classifier", *Neurocomputing*, v. 183, pp. 70–78. 4

CARDOSO, D. O., FRANÇA, F. M. G., GAMA, J., 2016b, "Clustering data streams using a forgetful neural model". In: Ossowski, S. (Ed.), *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*, pp. 949–951. ACM, b. 4, 4.4, 4.4.2

CARNEIRO, H. C. C., FRANÇA, F. M. G., LIMA, P. M. V., 2015, "Multilingual part-of-speech tagging with weightless neural networks", *Neural Networks*, v. 66, pp. 11–21. 1

CARPENTER, G. A., GROSSBERG, S., REYNOLDS, J. H., 1991, "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network", *Neural Networks*, v. 4, n. 5, pp. 565–588. 2.1.2

CARPENTER, G. A., GROSSBERG, S., MARKUZON, N., et al., 1992, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps", *IEEE Trans. Neural Networks*, v. 3, n. 5, pp. 698–713. 2.1.2

CARVALHO, D. S., CARNEIRO, H. C. C., FRANÇA, F. M. G., et al., 2013, "B-bleaching: Agile Overtraining Avoidance in the WiSARD Weightless Neural Classifier". In: *21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013*. 2.2.3

CHAKI, N., SHAIKH, S. H., SAEED, K., 2014, *Exploring Image Binarization Techniques*, v. 560, *Studies in Computational Intelligence*. Springer. 2.2.2

CHEN, C., ZHAN, Y., WEN, C., 2009, "Hierarchical Face Recognition Based on SVDD and SVM". In: *2009 International Conference on Environmental Science and Information Application Technology, ESIAT 2009, Wuhan, China, 4-5 July 2009, 3 Volumes*, pp. 692–695. 2.3.3

CHOW, C. K., 1970, "On optimum recognition error and reject tradeoff", *IEEE Trans. Information Theory*, v. 16, n. 1, pp. 41–46. 3.2

COUTINHO, P., CARNEIRO, H. C. C., CARVALHO, D. S., et al., 2014, "Extracting rules from DRASiW's "mental images"". In: *22th European Symposium on Artificial Neural Networks, ESANN 2014, Bruges, Belgium, April 23-25, 2014*. 1

DATAR, M., MOTWANI, R., 2016, "The Sliding-Window Computation Model and Results". In: Garofalakis, M. N., Gehrke, J., Rastogi, R. (Eds.), *Data Stream Management - Processing High-Speed Data Streams*, Data-Centric Systems and Applications, Springer, pp. 149–165. 2.4.2

DAWID, A. P., 1984, "Statistical theory: the prequential approach (with discussion)", *J. R. Statist. Soc. A*, v. 147, pp. 278–292. 2.4.3

DAY, W. H., EDELSBRUNNER, H., 1984, "Efficient algorithms for agglomerative hierarchical clustering methods", *Journal of classification*, v. 1, n. 1, pp. 7–24. 4.4

DE CARVALHO, D. S., FRANÇA, F. M. G., LIMA, P. M. V., 2014, "Extracting Semantic Information from Patent Claims Using Phrasal Structure Annotations". In: *2014 Brazilian Conference on Intelligent Systems, BRACIS 2014, Sao Paulo, Brazil, October 18-22, 2014*, pp. 31–36. IEEE Computer Society. 1

DENG, J., DONG, W., SOCHER, R., et al., 2009, "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. 3.3.3

FISCHER, L., HAMMER, B., WERSING, H., 2015, "Efficient rejection strategies for prototype-based classification", *Neurocomputing*, v. 169, pp. 334–342. 2.3.3

FONOLLOSA, J., SHEIK, S., HUERTA, R., et al., 2015, "Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring", *Sensors and Actuators B: Chemical*, v. 215, pp. 618 – 629. 4.4.2

FRENCH, R. M., 1999, "Catastrophic forgetting in connectionist networks", *Trends in Cognitive Sciences*, v. 3, n. 4, pp. 128 – 135. ISSN: 1364-6613. 2.1.2

FUMERA, G., ROLI, F., 2002, "Support Vector Machines with Embedded Reject Option". In: *Pattern Recognition with Support Vector Machines, First*

International Workshop, SVM 2002, Niagara Falls, Canada, August 10, 2002, Proceedings, pp. 68–82. 2.3.3

FUMERA, G., ROLI, F., GIACINTO, G., 2000, "Reject option with multiple thresholds", *Pattern Recognition*, v. 33, n. 12, pp. 2099–2101. 3.2, 3.2.2

GAMA, J., 2010, *Knowledge Discovery from Data Streams*. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press. ISBN: 978-1-4398-2611-9. 1, 4

GOUTTE, C., GAUSSIER, É., 2005, "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation". In: *Advances in Information Retrieval, 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005, Proceedings*, pp. 345–359. 3.2.2

GRANDVALET, Y., RAKOTOMAMONJY, A., KESHET, J., et al., 2008, "Support Vector Machines with a Reject Option". In: *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 537–544. 2.3.3

GRIECO, B. P. A., LIMA, P. M. V., GREGORIO, M. D., et al., 2010, "Producing pattern examples from "mental" images", *Neurocomputing*, v. 73, n. 7-9, pp. 1057–1064. 1, 2.2.3, 3, 4, 4

GRIFFIN, G., HOLUB, A., PERONA, P., 2007, *Caltech-256 Object Category Dataset*. Technical Report 7694, California Institute of Technology. 3.3.3

GROSSBERG, S., 1982, "How Does a Brain Build a Cognitive Code?" In: *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*, pp. 1–52, Dordrecht, Springer Netherlands. 2.1.2

HANCZAR, B., SEBAG, M., 2014, "Combination of One-Class Support Vector Machines for Classification with Reject Option". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I*, pp. 547–562. 2.3.3

HASSANI, M., SEIDL, T., 2016, "Using internal evaluation measures to validate the quality of diverse stream clustering algorithms", *Vietnam Journal of Computer Science*, pp. 1–13. ISSN: 2196-8896. doi: 10.

1007/s40595-016-0086-9. Available at: <http://dx.doi.org/10.1007/s40595-016-0086-9>. 4.4

HOMENDA, W., LUCKNER, M., PEDRYCZ, W., 2014, "Classification with rejection based on various SVM techniques". In: *2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014*, pp. 3480–3487. 2.3.3

HU, B., CHEN, Y., KEOGH, E. J., 2013, "Time Series Classification under More Realistic Assumptions". In: *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA.*, pp. 578–586. 3.3.2

IENCO, D., BIFET, A., ZLIOBAITE, I., et al., 2013, "Clustering Based Active Learning for Evolving Data Streams". In: Fürnkranz, J., Hüllermeier, E., Higuchi, T. (Eds.), *Discovery Science - 16th International Conference, DS 2013, Singapore, October 6-9, 2013. Proceedings*, v. 8140, *Lecture Notes in Computer Science*, pp. 79–93. Springer. 2.4.3

JAIN, L. P., SCHEIRER, W. J., BOULT, T. E., 2014, "Multi-class Open Set Recognition Using Probability of Inclusion". In: *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, pp. 393–409. 2.3.3, 3.3, 3.3.1

JIN, C., YU, J. X., ZHOU, A., et al., 2014, "Efficient Clustering of Uncertain Data Streams", *Knowl. Inf. Syst.*, v. 40, n. 3 (set.), pp. 509–539. ISSN: 0219-1377. 2.4.1, 4.4

JUNIOR, L. J. L., OLIVEIRA-SANTOS, T., BADUE, C., et al., 2015, "Image-based mapping, global localization and position tracking using VG-RAM weightless neural networks". In: *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pp. 3603–3610. IEEE. 2.2.2

KHAN, S. S., MADDEN, M. G., 2009, "A Survey of Recent Trends in One Class Classification". In: *Artificial Intelligence and Cognitive Science - 20th Irish Conference, AICS 2009, Dublin, Ireland, August 19-21, 2009, Revised Selected Papers*, pp. 188–197. 2.3

KOLCZ, A., ALLINSON, N., 1994, "Application of the CMAC input encoding scheme in the N-tuple approximation network", *Computers and Digital Techniques, IEE Proceedings* -, v. 141, n. 3 (May), pp. 177–183. 2.2.2

KRANEN, P., ASSENT, I., BALDAUF, C., et al., 2011, "The ClusTree: indexing micro-clusters for anytime stream mining", *Knowledge and Information Systems*, v. 29, n. 2, pp. 249–272. ISSN: 0219-1377. 2.4.2, 4.3, 4.4.2

LEVANDOWSKY, M., WINTER, D., 1971, "Distance between sets", *Nature*, v. 234, n. 5323, pp. 34–35. 4.3

LINNEBERG, C., JORGENSEN, T., 1999, "Discretization methods for encoding of continuous input variables for Boolean neural networks". In: *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, v. 2, pp. 1219–1224 vol.2, Jul. 2.2.2

LLOYD, S., 1982, "Least squares quantization in PCM", *IEEE Transactions on Information Theory*, v. 28, n. 2 (Mar), pp. 129–137. ISSN: 0018-9448. doi: 10.1109/TIT.1982.1056489. 4.4

LOWD, D., DOMINGOS, P. M., 2005, "Naive Bayes models for probability estimation". In: Raedt, L. D., Wrobel, S. (Eds.), *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, v. 119, *ACM International Conference Proceeding Series*, pp. 529–536. ACM. 3.1.3

LUDERMIR, T. B., CARVALHO, A., BRAGA, A., et al., 1999, "Weightless neural models: a review of current and past works", *Neural Computing Surveys*, v. 2, pp. 41–61. 2.2.3

MCCULLOCH, W. S., PITTS, W., 1943, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, v. 5, n. 4, pp. 115–133. 2.1.1, 2.1.3

MENA-TORRES, D., AGUILAR-RUIZ, J. S., 2014, "A similarity-based approach for data stream classification", *Expert Systems with Applications*, v. 41, n. 9, pp. 4224 – 4234. ISSN: 0957-4174. 2.1.2

MIROWSKI, P., LECUN, Y., 2012, "Statistical machine learning and dissolved gas analysis: a review", *Power Delivery, IEEE Transactions on*, v. 27, n. 4, pp. 1791–1799. 2.3.1, 3.3.1

NASCIMENTO, D., CARVALHO, R., MORA-CAMINO, F., et al., 2015, "A WiSARD-based multi-term memory framework for online tracking of objects". In: *23rd European Symposium on Artificial Neural Networks, ESANN 2015, Bruges, Belgium, April 22-24, 2015*. 2.2.2

OLIPHANT, T. E., 2007, "Python for Scientific Computing", *Computing in Science and Engineering*, v. 9, n. 3, pp. 10–20. 3.1.4

PAVLIDIS, N. G., TASOULIS, D. K., ADAMS, N. M., et al., 2011, "λ-Perceptronn: An Adaptive Classifier for Data Streams", *Pattern Recogn.*, v. 44, n. 1 (jan.), pp. 78–96. ISSN: 0031-3203. 2.1.2

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al., 2011, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, v. 12, pp. 2825–2830. 3.1.4, 3.3, 4.4

RODRIGUES, P. P., GAMA, J., 2009, "A system for analysis and prediction of electricity-load streams", *Intell. Data Anal.*, v. 13, n. 3, pp. 477–496. 2.1.2

ROSENBERG, A., HIRSCHBERG, J., 2007, "V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure". In: Eisner, J. (Ed.), *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pp. 410–420. ACL. 4.4

SAMET, H., 1995, "Spatial Data Structures". In: Kim, W. (Ed.), *Modern Database Systems: The Object Model, Interoperability, and Beyond.*, ACM Press and Addison-Wesley, pp. 361–385. 3.1.3

SCHEIRER, W. J., DE REZENDE ROCHA, A., SAPKOTA, A., et al., 2013, "Toward Open Set Recognition", *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 35, n. 7, pp. 1757–1772. 1, 2.3.2

SCHEIRER, W. J., JAIN, L. P., BOULT, T. E., 2014, "Probability Models for Open Set Recognition", *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 36, n. 11, pp. 2317–2324. 2.3.3

SCHÖLKOPF, B., PLATT, J. C., SHAWE-TAYLOR, J. C., et al., 2001, "Estimating the Support of a High-Dimensional Distribution", *Neural Comput.*, v. 13, n. 7 (jul.), pp. 1443–1471. 3.1.3

SEZGIN, M., SANKUR, B., 2004, "Survey over image thresholding techniques and quantitative performance evaluation", *J. Electronic Imaging*, v. 13, n. 1, pp. 146–168. 2.2.2

SIEGRIST, K., 1997. "The Birthday Problem". Available at: <http://www.math.uah.edu/stat/urn/Birthday.html>. Accessed: October 1, 2015. 3.1.2

SILVA, B., MARQUES, N. C., 2012, "Neural Network-based Framework for Data Stream Mining". In: Kersting, K., Toussaint, M. (Eds.), *STAIRS 2012 - Proceedings of the Sixth Starting AI Researchers' Symposium, Montpellier, France, 27-28 August 2012*, v. 241, *Frontiers in Artificial Intelligence and Applications*, pp. 294–305. IOS Press. 2.1.2

SILVA, J. A., FARIA, E. R., BARROS, R. C., et al., 2013, "Data Stream Clustering: A Survey", *ACM Comput. Surv.*, v. 46, n. 1 (jul.), pp. 13:1–13:31. ISSN: 0360-0300. 2.4.2

SOKOLOVA, M., LAPALME, G., 2009, "A systematic analysis of performance measures for classification tasks", *Inf. Process. Manage.*, v. 45, n. 4, pp. 427–437. 3.3.2

STAFFA, M., ROSSI, S., GIORDANO, M., et al., 2015, "Segmentation performance in tracking deformable objects via WNNs". In: *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pp. 2462–2467. IEEE. 2.2.2

TARLING, R., ROHWER, R., 1993, "Efficient use of training data in the n-tuple recognition method", *Electronics Letters*, v. 29, n. 24 (Nov), pp. 2093–2094. 2.2.3

TAX, D. M. J., DUIN, R. P. W., 2008, "Growing a multi-class classifier with a reject option", *Pattern Recognition Letters*, v. 29, n. 10, pp. 1565–1570. 2.3.3

VAN DER MAATEN, L. J. P., POSTMA, E. O., VAN DEN HERIK, H. J., 2007, *Dimensionality Reduction: A Comparative Review*. Technical report. 2.2.2

VINH, N. X., EPPS, J., BAILEY, J., 2010, "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance", *Journal of Machine Learning Research*, v. 11, pp. 2837–2854. 4.4

WAGNER, S., WAGNER, D., 2007. "Comparing Clusterings- An Overview". Available at: <http://staff.ustc.edu.cn/~zwp/teach/MVA/cluster_validation.pdf>. Accessed: October 15, 2016. 4.4

WAN, L., NG, W. K., DANG, X. H., et al., 2009, "Density-based Clustering of Data Streams at Multiple Resolutions", *ACM Trans. Knowl. Discov. Data*, v. 3, n. 3 (jul.), pp. 14:1–14:28. ISSN: 1556-4681. 2.4.2, 4.4

WAN, R., YAN, X., SU, X., 2008, "A Weighted Fuzzy Clustering Algorithm for Data Stream". In: *Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on*, v. 1, pp. 360–364, Aug. 2.4.2

WICKERT, I., FRANÇA, F. M. G., 2001, "AUTOWISARD: Unsupervised Modes for the WISARD". In: Mira, J., Prieto, A. (Eds.), *Connectionist Models of Neurons, Learning Processes and Artificial Intelligence, 6th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2001 Granada, Spain, June 13-15, 2001, Proceedings, Part I*, v. 2084, *Lecture Notes in Computer Science*, pp. 435–441. Springer. 1

XIANG, S., NIE, F., ZHANG, C., 2008, "Learning a Mahalanobis distance metric for data clustering and classification", *Pattern Recognition*, v. 41, n. 12, pp. 3600 – 3612. 3.1.3

ZAHN, C. T., 1971, "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters", *IEEE Trans. Computers*, v. 20, n. 1, pp. 68–86. 4.4.3

ZHANG, R., METAXAS, D. N., 2006, "RO-SVM: Support Vector Machine with Reject Option for Image Categorization". In: *Proceedings of the British Machine Vision Conference 2006, Edinburgh, UK, September 4-7, 2006*, pp. 1209–1218. 2.3.3

ZHU, Y., SHASHA, D. E., 2002, "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time". In: *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pp. 358–369. Morgan Kaufmann. 2.4.2

ZLIOBAITE, I., BIFET, A., PFAHRINGER, B., et al., 2014, "Active Learning With Drifting Streaming Data", *Neural Networks and Learning Systems, IEEE Transactions on*, v. 25, n. 1 (Jan), pp. 27–39. ISSN: 2162-237X. 2.4.1

List of Own Publications

CARDOSO, D. O., FRANÇA, F. M. G., GAMA, J. "Weightless Neural Modelling for Mining Data Streams". In: *Data Mining in Time Series and Streaming Databases*, World Scientific, pp. Accepted, to be published, 2017a.

BARBOSA, R., CARVALHO, D., FRANÇA, F. M. G., et al. "A neuro-symbolic approach to GPS trajectory classification". In: *ESANN 2017, 25th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 26-28, 2017, (Accepted, to be published)*, 2017.

CARDOSO, D. O., FRANÇA, F. M. G., GAMA, J. "WCDS: a Two-Phase Weightless Neural System for Data Stream Clustering", *New Generation Computing*, p. Under review, 2017b.

CARDOSO, D. O., GAMA, J., FRANÇA, F. M. G. "Weightless Neural Networks for Open Set Recognition", *Machine Learning*, p. Under review, 2017c.

CARDOSO, D. O., CARVALHO, D. S., ALVES, D. S. F., et al. "Financial credit analysis via a clustering weightless neural classifier", *Neurocomputing*, v. 183, pp. 70–78, 2016a.

CARDOSO, D. O., FRANÇA, F. M. G., GAMA, J. "Clustering data streams using a forgetful neural model". In: Ossowski, S. (Ed.), *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*, pp. 949–951. ACM, 2016b.

CARDOSO, D. O., FRANÇA, F. M. G., GAMA, J. "A bounded neural network for open set recognition". In: *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pp. 1–7. IEEE, 2015.

CARDOSO, D. O., CARVALHO, D. S., ALVES, D. S. F., et al. "Credit analysis with a clustering RAM-based neural classifier". In: *22th European*

Symposium on Artificial Neural Networks, ESANN 2014, Bruges, Belgium, April 23-25, 2014, 2014.

CARDOSO, D. O., GAMA, J., GREGORIO, M. D., et al. "WIPS: the WiSARD Indoor Positioning System". In: *21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013*, 2013.

ALVES, D., CARDOSO, D., CARNEIRO, H., et al. "An Empirical Study of the Influence of Data Structures on the Performance of VG-RAM Classifiers". In: *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI CBIC), 2013 BRICS Congress on*, pp. 388–393, Sept 2013.

CARDOSO, D. O., GREGORIO, M. D., LIMA, P. M. V., et al. "A Weightless Neural Network-Based Approach for Stream Data Clustering". In: *Intelligent Data Engineering and Automated Learning - IDEAL 2012 - 13th International Conference, Natal, Brazil, August 29-31, 2012. Proceedings*, pp. 328–335, 2012.

CARDOSO, D. O., LIMA, P. M. V., GREGORIO, M. D., et al. "Clustering data streams with weightless neural networks". In: *ESANN 2011, 19th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 27-29, 2011, Proceedings*, 2011.